MQL5 Dilinin REFERANSI

MetaTrader 5 müşteri terminali için

MQL5 dilini ÖĞRENİN ve her sorunu ÇÖZÜN:

- Herhangi bir karmaşıklık seviyesinde kendi teknik analiz göstergenizi oluşturun
- Otomatik alım-satım kullanın - çeşitli finansal piyasalar üzerinde çalışmak için alım-satım sisteminizi otomatik hale getirin
- Matematiksel gelişmelere veya geleneksel yöntemlere göre kendi analitik araçlarınızı oluşturun
- Geniş çaplı görevleri (alım-satım, görüntüleme, uyarı, vb.) çözebilmek için enformatik alım-satım sistemleri yazın

İçindekiler

MQL5 Referansı

MetaQuotes Language 5 (MQL5), finansal ticareti otomatikleştiren teknik göstergeleri, alım-satım robotlarını ve yardımcı uygulamaları geliştirmek için tasarlanan üst düzey bir dildir. MQL5 MetaQuotes tarafından kendi alım-satım platformu için geliştirmiştir. Sözdizimi, C++'a çok yakındır ve bu durum programcıların nesne yönelimli programlama (OOP) biçiminde uygulamalar geliştirmesini sağlar.

MQL5 diline ek olarak, alım-satım platformu paketi ayrıca şablonlar, kırpıntılar, hata ayıklama, profil çıkarma ve otomatik tamamlama araçları gibi yüksek düzeyde gelişmiş kod yazma araçlarına sahip MetaEditor IDE'nin yanı sıra, dosya versiyon belirlemeyi sağlayan yerleşik MQL5 Depolama'yı da içerir.

Dil desteği, büyük bir ücretsiz Kod Tabanı ve bol miktarda makale içeren MQL5.topluluk web sitesinde mevcuttur. Bu makaleler, sinir ağları, istatistik ve analiz, yüksek frekanslı alım-satım, arbitraj, ticaret stratejilerinin test ve optimizasyonu, alım-satım otomasyon robotlarının kullanımı ve daha fazlası dahil olmak üzere modern ticaretin tüm yönlerini kapsamaktadır.

Ticaret adamları and MQL5 program geliştiricileri forum üzerinde iletişim kurabilir, Freelance hizmetini kullanarak uygulama sipariş edebilir ve geliştirebilir, ayrıca otomatik ticaret uygulamaları mağaza'sında korumalı programlar satın alabilir ve satabilirler.

MQL5 dili, programcıların belirli işlem kurallarını takip ederek alım-satım işlemlerini otomatik olarak kontrol eden Uzman Danışmanları (EA) geliştirmelerine yardımcı olmak için özel işlem(alım-satım) fonksiyonlarını ve önceden tanımlanmış olay yöneticilerini sağlar. Uzman danışmanlara ek olarak, MQL5 özel teknik indikatörlerin, komut dosyalarının ve kütüphanelerin geliştirilmesine izin verir.

Bu MQL5 dili referansı, fonksiyonları, operasyonları, ayrılmış sözcükleri ve kategorilere ayrılmış diğer dil yapılarını içermektedir. Referans ayrıca alım-satım stratejileri, kontrol panelleri, özel grafikler geliştirmek ve dosya erişimini etkinleştirmek için kullanılan Standart Kütüphane sınıflarının açıklamalarını sağlamaktadır.

Ek olarak, Kod Tabanı, çeşitli matematik problemlerini çözmek için kullanılabilen ALGLIB sayısal analiz kütüphanesini içermektedir.

MQL5 Uygulama Türleri

MQL5 programları, uyguladıkları alım-satım otomasyon görevlerine dayalı olarak beş özelleşmiş türe ayrılır.

- **Uzman Danışman**, bir grafiğe bağlanmış otomatik bir alım-satım sistemidir. Bir Uzman Danışman, uygun alım-satım stratejisi elemanlarının gerçekleşimini etkinleştiren önceden tanımlanmış olayları yönetmek için olay yöneticileri içermektedir. Örneğin; bir programın başlatılması ve sonlandırılmasındaki olay, yeni fiyatlar, zamanlayıcı olayları, Piyasa Derinliğindeki değişiklikler, grafik olayları ve özel olaylar.
  Uygulanan kurallara dayanarak alım-satım sinyallerinin hesaplamasına ek olarak, Uzman Danışmanlar ayrıca otomatik olarak alım-satım işlemlerini gerçekleştirir ve onları direkt olarak bir alım-satım sunucusuna gönderir. Uzman danışmanlar, *<Terminal_Dizini>\MQL5\Experts* klasörü içerisinde saklanır.
- **Özel gösterge**, alım-satım platformuna entegre edilmiş var olan standart göstergelere ek olarak, bir kullanıcı tarafından geliştirilmiş bir teknik göstergedir. Standart olanlar hem de özel göstergeler otomatik olarak alım-satım işlemi yapamazlar, ancak sadece analitik fonksiyonlar uygulayabilirler. Özel göstergeler, hesaplamalarda diğer göstergelerin değerlerini kullanabilir ve uzman

danışmanlardan çağırılabilirler.
Özel göstergeler, <*Terminal_Dizini*>*MQL5**Indicators* klasörü içerisinde saklanır.

- **Komut dosyası**, bir eylemin tek bir gerçekleşimi için tasarlanmış bir programdır. Uzman danışmanların aksine, komut dosyaları; tetikleyici, başlatma ve sonlandırma olayları hariç hiç bir olayı yönetmezler. Bir komut dosyası kodu, OnStart yönetici fonksiyonunu içermelidir.
  Komut dosyaları, <*Terminal_Dizini*>*MQL5**Indicators klasörü içerisinde saklanır.*
- **Hizmet**; göstergelerin, Uzman Danışmanların ve komut dosyaların aksine çalışabilmesi için bir çizelgeye bağlı olmasına gerek olmayan bir programdır. Komut dosyaları gibi, hizmetler tetikleyici dışında hiçbir olayı yönetmezler. Bir hizmeti başlatmak için, kod OnStart işleyici fonksiyonunu içermelidir. Hizmetler, Başlat hariç başka herhangi bir olayı kabul etmez; ancak EventChartCustom'ı kullanarak grafiklere özel olaylar gönderebilirler. Hizmetler, <*Terminal_Dizini*>*MQL5**Services* klasörü içerisinde saklanır.
- **Kütüphane**, birtakım özel fonksiyonlardır. Kütüphaneler yaygın olarak kullanılan özel program algoritmalarını depolamak ve dağıtmak için tasarlanmıştır.
  Kütüphaneler, <*Terminal_Dizini*>*MQL5**Libraries* klasörü içerisinde saklanır.
- **İçerik dosyası**, sık kullanılan özel program bloklarının kaynak metnidir. Bu tür dosyalar derleme aşamasında Uzman Danışmanların, komut dosyalarının, özel göstergelerin ve kütüphanelerin kaynak metinlerine eklenebilir. Kütüphane fonksiyonları çağrılırken meydana gelen ek yük nedeniyle, içerik dosyalarının kullanımı kütüphanelerin kullanımından daha fazla tercih edilebilirdir.
  İçerik dosyaları orijinal dosyaların bulunduğu aynı dizinde saklanabilir. Bu durumda, çift tırnak ile #include yönergesi kullanılır. Diğer bir seçenek, içerik dosyalarını <*Terminal_Dizini*>*MQL5**Include* klasöründe saklamaktır. Bu durumda, açılı parantez ile #include yönergesi kullanılmalıdır.

Dil Temelleri

MetaQuotes Language 5 (MQL5), çeşitli finansal piyasaların analizi için, otomatikleştirilmiş alım-satım stratejilerinin ve özel teknik göstergelerin yazılması amacıyla geliştirilmiş, nesne yönelimli, yüksek seviyeli bir programlama dilidir. Gerçek zamanlı çalışma için tasarlanmış uzman sistemlerin yazımının yanında, alım-satım kararlarınıza yardımcı olan grafiksel araçların oluşturulmasına da izin verir.

MQL5, popüler programlama dili C++ konseptinden uyarlanmıştır. MQL4 ile karşılaştırıldığında yeni dil, sayımlar, yapılar, sınıflar ve olay işleme gibi kavramlara sahiptir. MQL5'deki çalıştırılabilir programlar ile diğer uygulamalar arasındaki dll aracılı etkileşim, gömülü ana tiplerin sayısının artırılması sayesinde artık olabildiğince kolaydır. MQL5 sözdizimi C++ sözdizimine benzer yapıdadır. Bu, diğer modern programlama dillerindeki programların çevrilmelerini kolaylaştırır.

MQL5 dili üzerinde çalışmanıza yardımcı olmak amacıyla tüm konular şu bölümler altında gruplanmıştır:

- Sözdizim
- Veri Tipleri
- İşlemler ve İfadeler
- Operatörler
- Fonksiyonlar
- Değişkenler
- Önişlemci
- Nesne Yönelimli Programlama
- Ad Alanları

Sözdizim

Alım-satım stratejilerinin programlaması için geliştirilmiş MQL5 dili, sözdizim açısından C++ programlama diline çok benzer, bazı özellikler dışında:

- adres aritmetiği yoktur;
- 'goto' operatörü yoktur;
- anonim bir sayım bildirilemez;
- çoklu kalıtım yoktur.

Ayrıca Bakınız

Sayımlar, Yapılar ve Sınıflar, Kalıtım

Yorumlar

Çok satırlı yorumlar '/*' sembol çifti ile başlar ve '*/' sembol çifti ile biter. Bu şekildeki yorumlar bitiştirilemez. Tek satır yorumlar '//' sembol çifti ile başlayıp yeni satır karakteri ile sona ererler, diğer çok satırlı yorumlar içinde bitiştirilebilirler. Yorumlar, boşluklara izin verilen her yerde olabilir ve içlerinde her sayıda boşluk bulunabilir.

Örnekler:

```
//--- Tek satırlık yorum
/*  Çok
   Satırlı      // iç içe geçmiş tek satırlık yorum
   yorum
*/
```

Tanımlayıcılar

Tanımlayıcılar değişken ismi veya fonksiyon isimi gibi kullanılırlar. Bir tanımlayıcının uzunluğu 63 karakteri geçemez.

Bir tanımlayıcıda kullanılmasına izin verilen karakterler: 0-9 şekilleri, büyük ve küçük latin harfler a-z ve A-Z (farklı karakterler şeklinde kabul edilirler), alt çizgi karakteri (_). İlk karakter sayı olamaz.

Tanımlayıcı, rezerve sözcüklerle aynı olamaz.

Örnekler:

```
NAME1 name1 Total_5 Paper
```

Ayrıca Bakınız

Değişkenler, Fonksiyonlar

Rezerve Sözcükler

Aşağıdaki tanımlayıcılar rezerve sözcükler olarak kaydedilmiştir, her biri belli bir eyleme karşılık gelir ve farklı bir anlamda kullanılamazlar:

Veri Tipleri

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

Erişim Belirteçleri

const	private	virtual
delete	protected	
override	public	

Bellek Sınıfları

extern	input	static

Operatörler

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

Diğer

this	#define	#import
true	#ifdef	#include
false	#ifndef	#property

this	#define	#import
template	#else	group
typename	#endif	namespace

Veri Tipleri

Her program veri ile çalışır. Veriler amaçlarına göre farklı tiplerde olabilirler. Örneğin tamsayı verileri dizi elemanlarına ulaşmak için kullanılır. Fiyat verisi, kayan noktalı çift duyarlık sayılarına aittir. Bu, MQL5 içinde fiyat verisi için özel bir tip oluşturulmamış olduğu gerçeğiyle alakalıdır.

Farklı tip veriler farklı hızlarla işlenirler. Tamsayı verileri en hızlı işlenenlerdir. Çift duyarlık sayılarını işlemek için, bir yardımcı işlemci kullanılır. Buna rağmen, verilerin kayan nokta ile temsilinin karmaşıklığı nedeniyle tamsayılardan yavaş işlenirler.

string tipli veriler, dinamik bellek tahsisi nedeniyle en yavaş işlenenlerdir.

Temel veri tipleri:

- Tamsayılar (char, short, int, long, uchar, ushort, uint, ulong);
- mantıksal (bool);
- harfli ifadeler (ushort);
- dizgiler (string);
- kayan nokta sayıları (double, float);
- renk (color);
- tarih ve zaman (datetime);
- sayımlar (enum).

Karmaşık veri tipleri:

- yapılar;
- sınıflar.

NYP terimlerinde karmaşık veri tipi, soyut veri tipi olarak geçer.

color ve *datetime* tipleri, sadece görselleştirme ve dışarıdan tanımlanan parametre girişi için uygundur (Uzman Danışmanlar tablosundan veya özel gösterge özelliklerindeki Veriler sekmesinden). *color* ve *datetime* tipleri tamsayı şeklinde temsil edilir. Tamsayı tipleri ve kayan nokta tipleri aritmetik (sayısal) tiplerdir.

Açık dönüşüm belirlenmemişse ifadelerde sadece gizli tip dönüşümü kullanılır

Ayrıca Bakınız

Tip Dönüşümü

Tamsayı Tipleri

MQL5 dilinde tamsayılar 11 tip ile temsil edilir. Program mantığı açısından gerekli olduğunda bazı tipler diğerleriyle birlikte kullanılabilir. Ama bu durumda tiplerin dönüşüm kurallarının hatırlanması gerekir.

Aşağıdaki tablo her bir tipin niteliklerini listelemektedir. Son sütun, her bir tip için C++ dilinde karşılık gelen tipi belirtmektedir.

Tip	Bayt Büyüklüğü	En Küçük Değer	En büyük değer	C++ Analog
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

Tamsayı tipli değerler; sayısal sabitler, renk kalıp-deyimleri, tarih-zaman kalıp-deyimleri, karakter sabitleri ve sayımlar şeklinde de temsil edilebilir.

Ayrıca Bakınız

Veri Dönüşümü, Nümerik Tip Sabitleri

Char, Short, Int ve Long Tipleri

char

char tipi 1 baytlık bellek işgal eder (8 bit) ve ikili notasyon (2^8=256) değerlerinin ifade edilmesini sağlar. *char* tipi negatif ve pozitif değerlerin her ikisini de barındırabilir. Değer aralığı -128'den 127'ye kadardır.

uchar

uchar tamsayı tipi de, *char* tipi gibi 1 baytlık bellek işgal eder; Ama onun aksine *uchar* sadece pozitif değerler için düşünülmüştür. En küçük değeri sıfırdır, en büyük değeri ise 255'tir. *uchar* tipinin isminin önündeki u harfi *unsigned* (işaretsiz) kelimesinin kısaltmasıdır.

short

short tipinin büyüklüğü 2 bayttır (16 bit) ve dolayısıyla, 2 üzeri 16'lık (2^16 = 65 536) bir aralıkta yer alan değerlerin ifade edilmesini sağlar. *short* tipinin işaretli bir tip olması ve hem negatif hem pozitif değerler içermesi nedeniyle, alabileceği değerler -32 768 ve 32 767 arasındadır.

ushort

İşaretsiz (unsigned) *short* tipi, *ushort* tipidir ve yine 2 bayt boyutundadır. En küçük değeri sıfır 0, En büyü değeri 65 535'tir.

int

Tamsayı *int* tipinin büyüklüğü 4 bayttır (32 bit). En küçük değer -2 147 483 648, en büyük değer ise 2 147 483 647'dir.

uint

İşaretsiz (unsigned) tamsayı tipi *uint*'tir. 4 baytlık belek işgal eder ve 0 - 4 294 967 295 arası tamsayıların ifade edilmesini sağlarlar.

long

long tipinin büyüklüğü 8 bayttır (64 bit). En küçük değer -9 223 372 036 854 775 808, en büyük değer ise 9 223 372 036 854 775 807'dir.

ulong

İşaretsiz *ulong* tipi de 8 bayt yer kaplar ve 0 ile 18 446 744 073 709 551 615 arası değerleri saklayabilirler.

Örnekler:

```
char  ch=12;
short sh=-5000;
```

```
int   in=2445777;
```

İşaretsiz tamsayı tipleri negatif değerler içerecek şekilde tasarlanmadıkları için negatif değer ile kullanımlarının denenmesi beklenmedik sonuçlar doğurur. Aşağıdaki gibi basit bir betik sonsuz bir döngüye yol açabilir:

```
//--- Sonsuz döngü
void OnStart()
  {
   uchar  u_ch;

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

Düzgün versiyon şu şekildedir:

```
//--- Düzgün versiyon şu şekildedir
void OnStart()
  {
   uchar  u_ch;

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

Sonuç:

```
   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
   ...
```

Örnekler:

```
//--- Negatif değerler, işaretsiz (unsigned) tiplerde saklanamazlar
uchar  u_ch=-120;
ushort u_sh=-5000;
uint   u_in=-401280;
```

Onaltılık: 0x veya 0X ile başlayan ve a-f/A-F harfleriyle (10-15 arası değerlerin yazımı için) ve 0-9 arası rakamlarla temsil edilirler.

Örnekler:

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

Tamsayı değişkenleri B ön eki kullanılarak ikili tabanda yazılabilir. Örneğin, bir alım-satım seansının çalışma saatlerini int tipli bir değişkene kodlayabilir ve ilgili bilgiyi istenen algoritmaya göre kullanabilirsiniz:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- çalışma saatleri için 1, diğer saatler içinse 0 değerini ayarla
   int AsianSession   =B'111111111'; // Asya seansı 0:00 - 9:00
   int EuropeanSession=B'111111111000000000'; // Avrupa seansı 9:00 - 18:00
   int AmericanSession =B'111111110000000000000011'; // Amerika seansı 16:00 - 02:00
//--- seansların sayısal değerlerini türet
   PrintFormat("Asya seans saatleri değeri =%d",AsianSession);
   PrintFormat("Avrupa seans saatleri değeri =%d",EuropeanSession);
   PrintFormat("Amerika seans saatleri değeri =%d",AmericanSession);
//--- şimdi, seansların çalışma saatlerini dizgi ifadeleri ile gösterelim
   Print("Asya seansı ",GetHoursForSession(AsianSession));
   Print("Avrupa seansı ",GetHoursForSession(EuropeanSession));
   Print("Amerika seansı ",GetHoursForSession(AmericanSession));
//---
  }
//+------------------------------------------------------------------+
//| seansların çalışma saatlerine bir dizgi şeklinde dönüş yap       |
//+------------------------------------------------------------------+
string GetHoursForSession(int session)
  {
//--- kontrol etmek için, AND bit işlemlerini ve 1 bitlik sağ kaydırmayı <<=1 kullan
//--- kontrole en düşük bitten başla
   int bit=1;
   string out="çalışma saatleri: ";
```

```
//--- sıfırdan yirmiüçe kadar artırarak 24 bitin tamamını kontrol et
   for(int i=0;i<24;i++)
     {
      //--- bit durumunu sayıyla al
      bool workinghour=(session&bit)==bit;
      //--- saati ifade eden sayıyı mesaja ekle
      if(workinghour )out=out+StringFormat("%d ",i);
      //--- bir sonrakinin değerini kontrol etmek için bir bit sola kaydır
      bit<<=1;
     }
//--- sonuç dizgisi
   return out;
  }
```

Ayrıca Bakınız

Tip Dönüşümü

Karakter Sabitleri

MQL5 dilindeki bir dizginin elemanları Unicode karakter kümesi içindeki işaretlerdir. Bunlar tamsayılara dönüştürülebilen ve toplama çıkarma gibi tamsayı işlemleriyle manüple edilebilen onaltılık değerlerdir.

Tırnak işaretleri içindeki her tekil karakter veya '\x10' gibi bir karakterin onaltılık ASCII kodu, bir karakter sabitidir ve ushort tipindedir. Örneğin '0' tipinde bir kaydın sayısal değeri 30'dur. Bu, karakterler tablosunda sıfır işaretinin indisine karşılık gelir.

Örnek:

```
void OnStart()
  {
//--- karakter sabitlerini tanımla
   int symbol_0='0';
   int symbol_9=symbol_0+9; // '9' sembolünü al
//--- sabitlerin çıktı değerleri
   printf("Ondalık biçimde: symbol_0 = %d,  symbol_9 = %d",symbol_0,symbol_9);
   printf("Onaltılık biçimde: symbol_0 = 0x%x,  symbol_9 = 0x%x",symbol_0,symbol_9);
//--- sabitleri bir dizgiye gir
   string test="";
   StringSetCharacter(test,0,symbol_0);
   StringSetCharacter(test,1,symbol_9);
//--- bir dizgide böyle görünürler
   Print(test);
  }
```

Ters-bölü (\) işareti, programın kaynak metnindeki sabit dizgilerle ve karakter sabitleriyle çalışırken kullanılan bir kontrol karakteridir. Bazı semboller (örneğin tek tırnak ('), çift tırnak ("), ters-bölü (\) ve kontrol karakterleri) ters-bölü işaretiyle (\) başlayan sembol kombinasyonları şeklinde, aşağıdaki tabloya göre temsil edilebilirler:

Karakter ismi	Hatırlatıcı kod veya simge	MQL5 içindeki kayıt	Nümerik değer
yeni satır (satır atlama)	LF	'\n'	10
yatay sekme	HT	'\t'	9
satır başı	CR	'\r'	13
ters-bölü	\	'\\'	92
tek tırnak	'	'\''	39
çift tırnak	"	'\"'	34
onaltılık kod	hhhh	'\xhhhh'	1 ile 4 arası onaltılık karakterler

Karakter ismi	Hatırlatıcı kod veya simge	MQL5 içindeki kayıt	Nümerik değer
ondalık kod	d	'\d'	0 ile 65535 arası ondalık sayılar

Ters-bölü işaretinin ardından, yukarıda tanımlanmayan şekilde bir karakter geliyorsa sonuç tanımsızdır.

Örnek

```
void OnStart()
  {
//--- karakter sabitlerini bildir
   int a='A';
   int b='$';
   int c='©';       // kod 0xA9
   int d='\xAE';    // ® sembolünün kodu
//--- çıktı sabitlerini çıktıla
   Print(a,b,c,d);
//--- dizgiye bir karakter ekle
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- dizgideki bir karakterin yerini değiştir
   StringSetCharacter(test,0,b);
   Print(test);
//--- dizgideki bir karakterin yerini değiştir
   StringSetCharacter(test,0,c);
   Print(test);
//--- dizgideki bir karakterin yerini değiştir
   StringSetCharacter(test,0,d);
   Print(test);
//--- karakteri bir sayı şeklinde ifade et
   int a1=65;
   int b1=36;
   int c1=169;
   int d1=174;
//--- dizgiye bir karakter ekle
   StringSetCharacter(test,1,a1);
   Print(test);
//--- dizgiye bir karakter ekle
   StringSetCharacter(test,1,b1);
   Print(test);
//--- dizgiye bir karakter ekle
   StringSetCharacter(test,1,c1);
   Print(test);
//--- dizgiye bir karakter ekle
   StringSetCharacter(test,1,d1);
```

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

Yukarıda bahsedildiği gibi, bir karakter sabitinin (veya değişkeninin) değeri, karakterler tablosundaki bir karakterin indisidir. Tamsayı değerli indis farklı yollarla yazılabilir.

```
void OnStart()
  {
//---
   int a=0xAE;      // ® işaretinin kodu '\xAE' sözcüğüne karşılık gelir
   int b=0x24;      // $ işaretinin kodu '\x24' sözcüğüne karşılık gelir
   int c=0xA9;      // © işaretinin kodu '\xA9' sözcüğüne karşılık gelir
   int d=0x263A;    // ☺ işaretinin kodu '\x263A' dizgisine karşılık gelir
//--- değerleri göster
   Print(a,b,c,d);
//--- dizgiye bir karakter ekle
   string test="";
   StringSetCharacter(test,0,a);
   Print(test);
//--- dizgideki bir karakterin yerini değiştir
   StringSetCharacter(test,0,b);
   Print(test);
//--- dizgideki bir karakterin yerini değiştir
   StringSetCharacter(test,0,c);
   Print(test);
//--- dizgideki bir karakterin yerini değiştir
   StringSetCharacter(test,0,d);
   Print(test);
//--- iskambil kartlarının kodları
   int a1=0x2660;
   int b1=0x2661;
   int c1=0x2662;
   int d1=0x2663;
//--- bir maça karakter ekle
   StringSetCharacter(test,1,a1);
   Print(test);
//--- bir kupa karakter ekle
   StringSetCharacter(test,2,b1);
   Print(test);
//--- bir karo karakter ekle
   StringSetCharacter(test,3,c1);
   Print(test);
//--- bir sinek karakter ekle
   StringSetCharacter(test,4,d1);
   Print(test);
//--- dizgi içindeki karakter sözcükleri örneği
   test="Queen\x2660Ace\x2662";
   printf("%s",test);
  }
```

Karakter sözcüklerinin içsel temsilleri ushort tipine sahiptir. Karakter sabitleri 0 ile 65535 arası değerler alabilir.

Ayrıca Bakınız

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

Datetime Tipi

datetime tipi, tarih ve zaman değerini saklamak için, 1 ocak 1970'den beri geçen saniyelerin sayısı şeklinde düşünülmüştür. Bu tip 8 baytlık bellek işgal eder.

Tarih ve zaman sabitleri, kelimelere ayrılmış bir dizgi şeklinde - yıl, ay, gün (veya gün, ay, yıl), saatler, dakikalar ve saniyelerin sayısal değerini gösteren altı kısımla temsil edilebilir. Sabit, tek tırnak işareti içine iliştirilmiştir ve D karakteri ile başlar.

Değer aralığı, 1 Ocak 1970'den 31 Aralık 3000'e kadardır. Tarih (yıl , ay , gün) veya zaman (saatler, dakikalar, saniyeler), veya tümü ihmal edilebilir.

Kelimelerle tarih belirlerken, yıl, ay ve gün belirtmeniz tavsiye edilir. Aksi durumda derleyici, tamamlanmamış girişle ilgili bir uyarı dönüşü yapacaktır.

Örnekler:

```
datetime NY=D'2015.01.01 00:00';      // 2015 yılının başlangıç zamanı
datetime d1=D'1980.07.19 12:30:27';   // Yıl Ay Gün Saatler Dakikalar Saniyeler
datetime d2=D'19.07.1980 12:30:27';   // D'1980.07.19 12:30:27' tarihine eşittir;
datetime d3=D'19.07.1980 12';         // D'1980.07.19 12:00:00' tarihine eşittir
datetime d4=D'01.01.2004';            // D'01.01.2004 00:00:00' tarihine eşittir
datetime compilation_date=__DATE__;              // Derleme tarihi
datetime compilation_date_time=__DATETIME__;     // Derleme tarihi ve zamanı
datetime compilation_time=__DATETIME__-__DATE__;// Derleme zamanı
//--- Derleyicinin uyarı dönmesine sebep olan bildiri örnekleri
datetime warning1=D'12:30:27';        // Eşittir D'[derleme tarihi] 12:30:27'
datetime warning2=D'';                // Eşittir __DATETIME__
```

Ayrıca Bakınız

Date Tipinin Yapısı, Tarih ve Zaman, TimeToString, StringToTime

Color Tipi

color tipi, renk hakkında bilgi depolamaya yöneliktir ve bellekte 4 baytlık yer kaplar. İlk bayt gözardı edilir, kalan 3 bayt ise RGB bileşenlerini içerir.

Renk sabitleri üç şekilde ifade edilebilir: harflerle, tamsayılarla veya isimlerle (sadece isimli Web renkleri).

Harflerle ifade, üç ana renk değişkeninin sayısal oran değerlerini temsil eden üç kısımdan oluşur: kırmızı, yeşil, mavi. Sabit, C ile başlar ve tırnak işaretlerinin arasında yer alır. Renk bileşenleri 0 ile 255 arasında değişen sayısal ağırlıklarla temsil edilirler.

Bu değerler onaltılık veya onluk sayılar biçiminde yazılır. Bir onaltılık sayı 0x00BBGGRR şeklinde görülür. RR kırmızı bileşen oranını, GG yeşil bileşen oranını ve BB mavi bileşen oranını simgeler. Onluk sabitler doğrudan RGB içine yansıtılmaz. Onaltılık tamsayı ifadesinin onluk değerini temsil eder.

Bazı renkler Web renkleri denilen ön-tanımlı renk kümesinden de seçilebilir.

Örnekler:

```
//--- harflerle
C'128,128,128'    // Gri
C'0x00,0x00,0xFF' // Mavi
//renk isimleri
clrRed               // Kırmızı
clrYellow            // Sarı
clrBlack             // Siyah
//--- tamsayı ifadeler
0xFFFFFF          // Beyaz
16777215          // Beyaz
0x008000          // Yeşil
32768             // Yeşil
```

Ayrıca Bakınız

Web Renkleri, ColorToString, StringToColor, Tip Dönüşümü

Bool Tipi

bool tipi, mantıksal true (doğru) veya false (yanlış) değerlerinin saklanması için düşünülmüştür. Bu değerlerin sayısal temsilleri sırasıyla 1 ve 0 şeklindedir.

Örnekler:

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

Mantıksal tipin içsel temsili 1 baytlık bütün bir sayıdır. Mantıksal ifadelerde, diğer tamsayıları veya reel tipleri veya bu tiplerin harfli ifadelerini kullanabileceğinizi lütfen not edin - derleyici herhangi bir hata oluşturmayacaktır. Bu durumda, sıfır değeri yanlış olarak, diğer tüm değerler ise doğru olarak yorumlanacaktır.

Örnekler:

```
   int i=5;
   double d=-2.5;
   if(i) Print("i = ",i," ve doğru olarak ayarlanır");
   else Print("i = ",i," ve yanlış olarak ayarlanır");

   if(d) Print("d = ",d," ve doğru değere sahip");
   else Print("d = ",d," ve yanlış değere sahip");

   i=0;
   if(i) Print("i = ",i," ve doğru değere sahip");
   else Print("i = ",i," ve yanlış değere sahip");

   d=0.0;
   if(d) Print("d = ",d," ve doğru değere sahip");
   else Print("d = ",d," ve yanlış değere sahip");

//--- Çalıştırma sonuçları
//   i= 5 ve doğru değere sahip
//   d= -2.5 ve doğru değere sahip
//   i= 0 ve yanlış değere sahip
//   d= 0 ve yanlış değere sahip
```

Ayrıca Bakınız

Mantıksal İşlemler, Öncelik Kuralları

Sayımlar

enum tipli veriler belirli ve sınırlı bir veri kümesine aittir. Sayım tipi şu şekilde tanımlanabilir:

```
enum sayılabilir tipin ismi
  {
  değerlerin listesi
  };
```

Değerler listesi, isimlendirilmiş sabitlerin virgülle ayrılmış bir listesidir.

Örnek:

```
enum months  // isimlendirilmiş sabitlerin sayımı
   {
    January,
    February,
    March,
    April,
    May,
    June,
    July,
    August,
    September,
    October,
    November,
    December
   };
```

Sayım bildirildikten sonra tamsayı değerli 4-baytlık yeni bir veri tipi görünür. Yeni veri tipinin bildirimi, derleyicinin geçirilen parametreleri sıkıca kontrol etmesini sağlar; çünkü sayım, yeni isimlendirilmiş sabitleri tanıtır. Yukarıdaki örnekte, January isimli sabit 0 değerine, February - 1 değerine ve December - 11 değerine sahiptir.

Kural: Sayımdaki sabitlerden biri için belirli bir değer atanmamışsa, ilgili değer otomatik olarak derleyici tarafından verilir. Eğer bu sabit sayımın ilk üyesiyse değeri '0' olarak ayarlanır. Takip eden tüm üyeler için değerler, bir önceki değerin üstüne bir eklenerek hesaplanacaktır.

Örnek:

```
enum intervals  // İsimlendirilen sabitlerin sayımı
   {
    month=1,     // Bir aylık aralık
    two_months,  // İki ay
    quarter,     // Üç ay - çeyrek
    halfyear=6,  // Yarım yıl
    year=12,     // Yıl - 12 ay
   };
```

Notlar

- C++ dilinin aksine, sayım tipinin içsel temsilinin MQL5'deki büyüklüğü, her zaman 4 bayta eşittir. Yani sizeof (months), 4 değerine dönüş yapar.
- C++ dilinin aksine, MQL5'de bir anonim sayım bildirilemez. Yani enum anahtar sözcüğünden sonra, her zaman benzersiz isim isim belirtilmelidir.

Ayrıca Bakınız

Tip Dönüşümü

Reel Tipler (double, float)

Reel tipler (yada kayan noktalı tipler) ondalık kısımları olan değerleri temsil eder. MQL5 dilinde kayan nokta sayıları için iki farklı tip mevcuttur. Bilgisayar hafızasında reel sayıların temsil yöntemi IEEE 754 standardı ile tanımlanır ve platformlardan, işletim sistemlerinden ve programlama dillerinden bağımsızdır.

Tip	Bayt bazında büyüklük	Minimal Pozitif Değer	En büyük değer	C++ Analog
float	4	1.175494351e-38	3.402823466e+38	float
double	8	2.2250738585072014e-308	1.7976931348623158e+308	double

double

double gerçek sayı türü 64 bit (1 işaret biti, 11 üs biti ve 52 mantis biti) kaplar.

float

float gerçek sayı türü 32 bit (1 işaret biti, 8 üs biti ve 23 mantis biti) kaplar.

vector

double türünde sayılardan oluşan tek boyutlu dizidir. Veriler için bellek dinamik olarak tahsis edilir. Metotlar kullanılarak vektör özellikleri elde edilebilir ve vektör büyüklüğü değiştirilebilir. Şablon fonksiyonlarında vector<double> girdisi kullanılabilir.

vectorf

Hassasiyet kaybı önemli değilse, vector yerine float türünde sayılardan oluşan tek boyutlu dizi olan vectorf kullanılabilir. Şablon fonksiyonlarında vector<float> girdisi kullanılabilir.

vectorc

complex türünde sayılardan oluşan tek boyutlu dizidir. Karmaşık sayıların işlenmesi içindir. Şablon fonksiyonlarında vector<complex> girdisi kullanılabilir. vectorc türündeki vektörler üzerinde işlemler henüz tanımlanmamıştır.

matrix

double türünde sayılardan oluşan iki boyutlu dizidir. Veriler için bellek dinamik olarak tahsis edilir. Metotlar kullanılarak matris özellikleri elde edilebilir ve matris şekli değiştirilebilir. Şablon fonksiyonlarında matrix<double> girdisi kullanılabilir.

matrixf

Hassasiyet kaybı önemli değilse, matrix yerine float türünde sayılardan oluşan iki boyutlu dizi olan matrixf kullanılabilir. Şablon fonksiyonlarında matrix<float> girdisi kullanılabilir.

matrixc

complex türünde sayılardan oluşan iki boyutlu dizidir. Karmaşık sayıların işlenmesi içindir. Şablon fonksiyonlarında matrix<complex> girdisi kullanılabilir. matrixc türündeki vektörler üzerinde işlemler henüz tanımlanmamıştır.

double ismi, bu sayıların çözünürlüğünün float tipli sayıların iki katı olmasından gelir. Çoğu durumda double tipi en uygun olandır. Çoğu durumda ise float sayıların sınırlı çözünürlükleri yetersizdir. float tipinin hala kullanılmasının sebebi bellek tasarrufudur (bu, geniş reel sayı dizileri için önemlidir).

Kayan noktalı sabitler bir tamsayı kısmı, bir nokta (.) ve bir ondalık kısımdan oluşur. Tamsayı ve kesirli kısımlar, ondalık sayı dizileridir.

Örnekler:

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

Reel sabitleri yazmanın bilimsel bir yolu vardır ve bu kayıt yöntemi çoğunlukla geleneksel yöntemden daha derli topludur.

Örnek:

```
double c1=1.12123515e-25;
double c2=0.00000000000000000000000112123515; // noktadan sonra 24 sıfır

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

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

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

Reel sayıların ondalık notasyon kullanılırken bellek üzerinde ikili sistemde ve sınırlı kesinlikle depolandığı hatırlanmalıdır. Bu, ondalık sistemde tam olarak temsil edilen birçok sayının, ikili sistemde sadece sonsuz bir kesir olarak yazılmasını açıklar.

Örneğin, 0.3 ve 0.7 sayıları, bilgisayarda sonsuz bölümlerle temsil edilirken, 0.25 tam olarak saklanır, çünkü ikinin kuvvetini temsil eder.

Bu yüzden, iki reel sayının eşitliklerinin kontrolü için karşılaştırılma yapılması tavsiye edilmez; böyle bir karşılaştırma doğru değildir.

Örnek:

```
void OnStart()
  {
//---
   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");
// Sonuç: 1/3 + 4/3 != 5/3
  }
```

Eğer hala iki reel sayıyı karşılaştırmak istiyorsanız bunu iki şekilde yapabilirsiniz. İlk yol, iki sayı arasındaki farkı karşılaştırmanın doğruluğunu belirleyecek küçük bir meblağ ile karşılaştırmaktır.

Örnek:

```
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;
  }
void OnStart()
  {
   double d_val=0.7;
   float  f_val=0.7;
   if(EqualDoubles(d_val,f_val,0.000000000000001))
      Print(d_val," eşittir ",f_val);
   else
      Print("Farklı: d_val = ",DoubleToString(d_val,16),"  f_val = ",DoubleToString(f_
// Sonuç: Farklı: d_val= 0.7000000000000000   f_val= 0.6999999880790710
  }
```

Yukarıdaki örnekteki epsilon değerinin, ön tanımlı sabit DBL_EPSILON değerinden küçük olamayacağını not edin. Bu sabitin değeri 2.2204460492503131e-016 sayısına eşittir. Float tipine karşılık gelen sabit

FLT_EPSILON = 1.192092896e-07 sayısıdır. Bunlar şu anlama gelir: Bu değer, 1.0 + DBL_EPSILON koşulunu sağlayan en düşük değerdir! = 1.0 (float tipli sayılar için 1.0 + FLT_EPSILON! = 1.0).

İkinci yol, iki reel sayının normalleştirilmiş farklarını sıfırla karşılaştırmayı önerir. Normalleştirilmiş farkın sıfır ile karşılaştırılması anlamsızdır. Normalleştirilmiş sayılarla yapılan tüm matematiksel işlemler normalize olmayan sonuçlar verirler.

Örnek:

```
bool CompareDoubles(double number1,double number2)
  {
   if(NormalizeDouble(number1-number2,8)==0)
      return(true);
   else
      return(false);
  }
void OnStart()
  {
   double d_val=0.3;
   float  f_val=0.3;
   if(CompareDoubles(d_val,f_val))
      Print(d_val," equals ",f_val);
   else
      Print("Farklı: d_val = ",DoubleToString(d_val,16),"  f_val = ",DoubleToString(f_
// Sonuç: Farklı: d_val= 0.3000000000000000   f_val= 0.3000000119209290
  }
```

Matematiksel yardımcı işlemcinin bazı işlemleri geçersiz reel sayı ile sonuçlanabilir. Böyle bir sonuç matematiksel işlemlerde ve karşılaştırmalarda kullanılamaz, çünkü geçersiz reel sayılarla yapılan işlemlerin sonucu tanımsızdır. Örneğin, arcsin(2) değerini hesaplamaya çalıştığımızda sonuç negatif sonsuz olur.

Örnek:

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

Eksi sonsuzun yanında, artı sonsuz ve NaN (not a number: bir sayı değil) bulunmaktadır. Sayının geçersiz olup olmadığını öğrenmek için MathIsValidNumber() fonksiyonunu kullanabilirsiniz. IEEE standardına göre, bunlar özel makine temsiline sahiptirler. Örneğin, double tip için artı sonsuz değeri 0x7FF0 0000 0000 0000 şeklinde bit temsiline sahiptir.

Örnekler:

```
struct str1
  {
   double d;
  };
struct str2
  {
   long l;
```

```
  };

//--- Başlat
   str1 s1;
   str2 s2;
//---
   s1.d=MathArcsin(2.0);         // -1.#IND geçersiz sayısını al
   s2=s1;
   printf("1.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFFF000000000000;      // geçersiz sayı -1.#QNAN
   s1=s2;
   printf("2.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF7000000000000;      // en büyük sayı-olmayan SNaN
   s1=s2;
   printf("3.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF8000000000000;      // en küçük sayı-olmayan QNaN
   s1=s2;
   printf("4.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FFF000000000000;      // en büyük sayı-olmayan QNaN
   s1=s2;
   printf("5.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0x7FF0000000000000;      // Artı sonsuz 1.#INF ve en küçük sayı-olmayan SNaN
   s1=s2;
   printf("6.   %f %I64X",s1.d,s2.l);
//---
   s2.l=0xFFF0000000000000;      // Eksi sonsuz -1.#INF
   s1=s2;
   printf("7.  %f %I64X",s1.d,s2.l);
//---
   s2.l=0x8000000000000000;      // Negatif sıfır -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;      // En büyük normalize sayı (MAX_DBL)
   s1=s2;
   printf("11.  %.16e %I64X",s1.d,s2.l);
```

```
//---
   s2.l=0x0010000000000000;      // En küçük normalize sayı (MIN_DBL)
   s1=s2;
   printf("12.  %.16e %.16I64X",s1.d,s2.l);
//---
   s1.d=0.7;                     // 0.7 sayısını göster - sonsuz fraksiyon
   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
*/
```

Ayrıca Bakınız

DoubleToString, NormalizeDouble, Nümerik Tipli Sabitler

Karmaşık sayı (complex)

Yerleşik complex tür, iki double alana sahip bir yapıdır:

```
struct complex
  {
   double            real;    // Reel kısım
   double            imag;    // Sanal kısım
  };
```

"complex" tipi, MQL5 fonksiyonları için bir parametre olarak değer cinsinden iletilebilir (sadece referans ile geçişi yapılan sıradan yapıların aksine). DLL'lerden içe aktarılan fonksiyonlar için "complex" tipinin yalnızca referans yoluyla geçişi yapılabilir.

'i' takısı karmaşık sabitleri tanımlamak için kullanılır:

```
complex square(complex c)
  {
   return(c*c);
  }
void OnStart()
  {
   Print(square(1+2i));  // parametre olarak bir sabit iletilir
  }
// karmaşık sayının bir dizge temsili olan "(-3,4)" sıktısı alınacaktır.
```

Şu anda karmaşık sayılar için yalnızca basit işlemler mevcuttur: =, +, -, *, /, +=, -=, *=, /=, ==,!=.

Gelecekte, ek matematiksel işlevler eklenecektir: mutlak değerin elde edilmesi, sinüs, kosinüs vb.

vectorc

complex türünde sayılardan oluşan tek boyutlu dizidir. Karmaşık sayıların işlenmesi içindir. Şablon fonksiyonlarında vector<complex> girdisi kullanılabilir. vectorc türündeki vektörler üzerinde işlemler henüz tanımlanmamıştır.

matrix

double türünde sayılardan oluşan iki boyutlu dizidir. Veriler için bellek dinamik olarak tahsis edilir. Metotlar kullanılarak matris özellikleri elde edilebilir ve matris şekli değiştirilebilir. Şablon fonksiyonlarında matrix<double> girdisi kullanılabilir.

matrixf

Hassasiyet kaybı önemli değilse, matrix yerine float türünde sayılardan oluşan iki boyutlu dizi olan matrixf kullanılabilir. Şablon fonksiyonlarında matrix<float> girdisi kullanılabilir.

matrixc

complex türünde sayılardan oluşan iki boyutlu dizidir. Karmaşık sayıların işlenmesi içindir. Şablon fonksiyonlarında matrix<complex> girdisi kullanılabilir. matrixc türündeki vektörler üzerinde işlemler henüz tanımlanmamıştır.

String Tipi

string tipi, metin dizgilerini saklamak için kullanılır. Metin dizgileri sonu sıfırlı Unicode biçimli karakterlerden oluşan söz dizeleridir. Dizgi sabitleri string tipli değişkenlere atanabilir. Dizgi sabitleri, tırnak içine iliştirilmiş Unicode karakter dizeleridir: "Bu bir dizgi sabitidir".

Eğer bir dizginin içine tırnak işareti (") eklemek istiyorsanız, bunun öncesinde (\) karakterini kullanmanız gerekir. Tüm özel karakter sabitleri (\) karakteri kullanılarak bir dizgi içine yazılabilir.

Örnekler:

```
string svar="Bu bir karakter dizesi";
string svar2=StringSubstr(svar,0,4);
Print("Copyright symbol\t\x00A9");
FileWrite(handle,"Bu dizgi, yeni satır \n sembolleri içeriyor ");
string MT5path="C:\\Program Files\\MetaTrader 5";
```

Kodun okunabilirliğini artırmak amacıyla, uzun sabit dizgiler toplama işlemi olmaksızın parçalara bölünebilirler. Derleme sırasında bu parçalar tek bir uzun dizgide birleştirilecektir:

```
//--- Uzun bir sabit dizgi bildir
   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>Trade Operations Report</title>\n"
                    "</head>";
//--- Sabit dizgiyi, günlüğe çıktıla
   Print(HTML_head);
  }
```

Ayrıca Bakınız

Dönüşüm Fonksiyonları, Dizgi Fonksiyonları, FileOpen, FileReadString, FileWriteString

Yapılar, Sınıflar ve Arayüzler

Yapılar

Yapılar her tipten elemanlar içerebilen kümelerdir (void tipi dışında). Yapıların kullanımı mantıksal olarak ilişkili farklı tip verilerin bir araya getirilmesini sağlar.

Yapı Bildirimi

Yapı veri tipi şu tarif ile belirtilir:

```
struct yapı_ismi
  {
  elemanların_açıklaması
  };
```

Yapı ismi bir tanımlayıcı (bir değişken veya fonksiyon ismi) şeklinde kullanılamaz. MQL5 yapı elemanlarının sıralama olmaksızın doğrudan birbirlerini takip ettiği not edilmelidir. Bu gibi bir sıralama C++ dilinde şu yönerge ile derleyiciye yaptırılır:

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

Eğer yapı içinde başka bir sıralama yapmak isterseniz doğru ölçüde yardımcı üyeler ve "dolgular" kullanın.

Örnek:

```
struct trade_settings
  {
   uchar  slippage;     // izin verilebilir slipaj (sapma) değeri 1 bayt
   char   reserved1;    // 1 bayt atla
   short  reserved2;    // 2 bayt atla
   int    reserved4;    // başka bir 4 bayt atlandı. 8 baytlık sınır hizasını temin ed
   double take;         // kar sabitleme fiyatının değerleri
   double stop;         // koruyucu durdurma seviyesinin fiyat değeri
  };
```

Sıralanmış yapıların bu tanımı, sadece içe aktarılmış dll fonksiyonlarına yapılacak veri aktarımları için gereklidir.

Uyarı: Bu örnek yanlış tasarlanmış veriler sergiler. Önce double tipli *take* ve *stop* gibi büyük değişkenlerin, ardından uchar tipli *slippage* üyesinin bildirilmesi daha iyi olacaktır. Bu durumda, verinin içsel temsili *#pragma pack()* içinde belirlenen değerden bağımsız olarak her zaman aynı olacaktır.

Eğer bir yapı string tipi ve veya dinamik dizi nesnesi şeklinde değişkenler içeriyorsa, derleyici yapı için gizli bir yapıcı fonksiyon atar. Bu yapıcı tüm string tipli üyeleri sıfırlar ve dinamik dizi nesnelerini düzgün şekilde başlatır.

Basit Yapılar

Dizgileri, nesne işaretçilerini, sınıf nesnelerini ve dinamik dizileri içermeyen yapılar basit yapılardır. Yapı değişkenleri ve bunların dizileri DLL dosyalarından aktarılan fonksiyonlara geçirilebilir.

Basit yapıların kopyalanmasına sadece iki durumda izin verilir:

- Nesne aynı yapı tipinden ise
- Nesnelerin köken bağı varsa, yani bir yapı diğerinden türetilmişse.

Örnek olarak, gömülü MqlTick yapısını kullanarak, aynı içeriğe sahip olan özel CustomMqlTick yapısını oluşturalım. Bu durumda derleyici MqlTick nesne değerinin CustomMqlTick tipli nesneye kopyalanmasına izin vermeyecektir. Uygun tipe yapılan doğrudan tip dönüşümü bile derleyici hatası verecektir:

```
      //--- basit yapılar farklı tiplere dönüştürülemez
      my_tick1=last_tick;               // burada derleyici hata dönüşü yapar

      //--- farklı tipli yapıların birbirlerinin tipine dönüştürülmesi de mümkün değil
      my_tick1=(CustomMqlTick)last_tick;// burada derleyici hata dönüşü yapar
```

Bu nedenle tek bir seçenek kalır, o da yapı elemanlarının değerlerini tek tek kopyalamaktır. CustomMqlTick ile aynı tipte olan değerlerin kopyalanması mümkündür.

```
      CustomMqlTick my_tick1,my_tick2;
      //--- CustomMqlTick ile aynı tipteki verilerin kopyalanması şu şekilde mümkündür
      my_tick2=my_tick1;

      //--- CustomMqlTick yapısının nesnelerinden bir dizi oluştur ve değerleri buna y
      CustomMqlTick arr[2];
      arr[0]=my_tick1;
      arr[1]=my_tick2;
```

ArrayPrint() fonksiyonu *arr[] dizisinin değerini günlükte görüntülemek için* çağrılır.

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- gömülü MqlTick yapısına benzer bir yapı oluştur
   struct CustomMqlTick
     {
      datetime          time;          // Son fiyatın güncelleme zamanı
      double            bid;           // Mevcut Satış fiyatı
      double            ask;           // Mevcut Alış fiyatı
      double            last;          // Son işlemin mevcut fiyatı
      ulong             volume;        // Son işlemin hacmi
      long              time_msc;      // Son işlemin güncelleme zamanı
      uint              flags;         // Tik bayrakları
     };
   //--- son tik fiyatını al
   MqlTick last_tick;
   CustomMqlTick my_tick1,my_tick2;
```

```
//--- MqlTick verilerini CustomMqlTick üzerine kopyalama denemesi
   if(SymbolInfoTick(Symbol(),last_tick))
     {
      //--- ilgisiz basit yapıların kopyalanmasına izin verilmez
      //1. my_tick1=last_tick;               // derleyici burada bir hata dönüşü yapar

      //--- ilgisiz yapılar için tip dönüşümüne de izin verilmez
      //2. my_tick1=(CustomMqlTick)last_tick;// derleyici burada bir hata dönüşü yapar

      //--- yani, yapı elemanlarını tek tek kopyalamalıyız
      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;

      //--- CustomMqlTick ile aynı tipteki verilerin kopyalanması şu şekilde mümkündür
      my_tick2=my_tick1;

      //--- CustomMqlTick yapısının nesnelerinden bir dizi oluştur ve değerleri buna y
      CustomMqlTick arr[2];
      arr[0]=my_tick1;
      arr[1]=my_tick2;
      ArrayPrint(arr);
//--- CustomMqlTick tipli nesneler içeren dizinin değerlerinin görüntülenmesi için örn
      /*
                         [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() başarısız, hata = ",GetLastError());
  }
```

İkinci örnek aynı soydan iki basit yapının kopyalanmasını göstermektedir. Elimizde Animal (hayvan) yapısından türetilen Cat (kedi) ve Dog (köpek) gibi iki basit yapı olsun. Animal ve Cat nesnelerini (aynı şekilde Animal ve Dog nesnelerini) birbirine kopyalayabiliriz. Ama Cat ve Dog nesneleri aynı soydan (Animal) gelmelerine rağmen birbirlerine kopyalanamaz.

```
//--- köpekleri tanımlayan yapı
struct Dog: Animal
  {
   bool              hunting;       // vahşi tür
  };
//--- kedileri tanımlayan yapı
struct Cat: Animal
  {
```

```
   bool              home;          // evcil tür
  };
//--- türetik yapıları oluştur
   Dog dog;
   Cat cat;
//--- atadan neslinde olana kopyalama yapılabilir (Animal ==> Dog)
   dog=some_animal;
   dog.swim=true;    // köpekler yüzebilir
//--- türetik yapılar birbirlerine kopyalanamaz (Dog != Cat)
   cat=dog;         // derleyici hata dönüşü yapar
```

Örneğin tam kodu:

```
//--- hayvanları tanımlayan temel yapı
struct Animal
  {
   int               head;          // baş sayısı
   int               legs;          // bacak sayısı
   int               wings;         // kanat sayısı
   bool              tail;          // kuyruk
   bool              fly;           // uçucu
   bool              swim;          // yüzücü
   bool              run;           // koşucu
  };
//--- köpekleri tanımlayan yapı
struct Dog: Animal
  {
   bool              hunting;       // vahşi tür
  };
//--- kedileri tanımlayan yapı
struct Cat: Animal
  {
   bool              home;          // evcil tür
  };
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- temel Animal (hayvan) tipli nesneyi tanımla
   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;
//--- türetik nesne tiplerini oluştur
```

```
   Dog dog;
   Cat cat;
//--- atadan neslinde olana kopyalama yapılabilir (Animal ==> Dog)
   dog=some_animal;
   dog.swim=true;    // köpekler yüzebilir
//--- türetik yapılar birbirlerine kopyalanamaz (Dog != Cat)
   //cat=dog;        // derleyici burada hata dönüşü yapar
//--- ama nesne bileşenleri tek tek kopyalanabilir
   cat.head=dog.head;
   cat.legs=dog.legs;
   cat.wings=dog.wings;
   cat.tail=dog.tail;
   cat.fly=dog.fly;
   cat.swim=false;   // kediler yüzemez
//--- neslinde olandan ataya kopyalama yapılabilir
   Animal elephant;
   elephant=cat;
   elephant.run=false;// filler koşamaz
   elephant.swim=true;// filler yüzer
//--- Bir dizi oluştur
   Animal animals[4];
   animals[0]=some_animal;
   animals[1]=dog;
   animals[2]=cat;
   animals[3]=elephant;
//--- sonuçları çıktıla
   ArrayPrint(animals);
//--- çalıştırma sonucu
/*
        [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
*/
  }
```

Basit tipli yapıları kopyalamanın bir diğer yolu bileşimdir. Yapı elemanları aynı bileşimin üyeleri olmalıdır - bkz. bileşim örneği.

Yapı Üyelerine Erişim

Yapılara verilen isimler yeni veri tiplerini temsil eder ve yeni değişkenlerin bildirimi bu tipler ile yapılabilir. Yapılar bir proje içinde yalnızca bir defa bildirilebilir. Yapı üyelerine nokta işlemi (.) kullanılarak erişilebilir.

Örnek:

```
struct trade_settings
  {
```

```
   double take;          // kar sabitleme değerleri
   double stop;          // koruyucu durdurma fiyatı değeri
   uchar  slippage;      // kabul edilebilir slipaj
  };
//--- trade_settings tipinde bir değişken oluştur ve başlat
trade_settings my_set={0.0,0.0,5};
if (input_TP>0) my_set.take=input_TP;
```

yapı ve sınıf alanlarını hizalamak için 'pack'

Özel pack özelliği, yapı veya sınıf alanlarının hizalanmasını sağlar.

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

burada n, şu değerlerden biridir: 1, 2, 4, 8 ya da 16. n değeri bulunmayadabilir.

Örnek:

```
   struct pack(sizeof(long)) MyStruct
     {
      // yapı üyeleri 8 bayt sınırına hizalanacak
     };
or
   struct MyStruct pack(sizeof(long))
     {
      // yapı üyeleri 8 bayt sınırına hizalanacak
     };
```

Yapılar için varsayılan olarak 'pack(1)' uygulanır. Bu, yapı elemanlarının birbiri ardına hafızaya yerleştirildiği ve yapı büyüklüğünün üyelerinin büyüklüğünün toplamına eşit olduğu anlamına gelir.

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- hizasız basit yapı
   struct Simple_Structure
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
   //--- basit bir yapı örneği bildir
   Simple_Structure s;
//--- her yapı üyesinin boyutunu göster
   Print("sizeof(s.c)=",sizeof(s.c));
   Print("sizeof(s.s)=",sizeof(s.s));
```

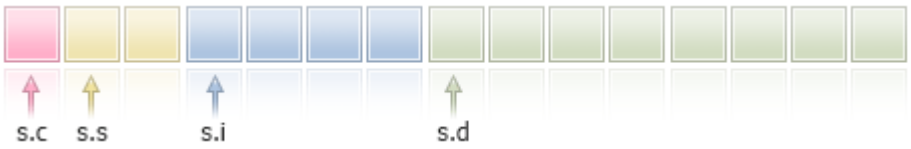
```
   Print("sizeof(s.i)=",sizeof(s.i));
   Print("sizeof(s.d)=",sizeof(s.d));
//--- POD yapısının büyüklüğünün, üyelerinin büyüklüğünün toplamına eşit olduğundan em
   Print("sizeof(simple_structure)=",sizeof(simple_structure));
/*
  Result:
   sizeof(s.c)=1
   sizeof(s.s)=2
   sizeof(s.i)=4
   sizeof(s.d)=8
   sizeof(simple_structure)=15
*/
  }
```

Bu tür bir hizalamanın uygulandığı üçüncü taraf kütüphaneleriyle (*.DLL) veri alışverişinde bulunurken yapı alanlarının hizalanması gerekebilir.

Hizalamanın nasıl çalıştığını göstermek için bazı örnekler kullanalım. Dört üyeden oluşan hizasız bir yapı uygulayacağız.

```
//--- hizasız basit yapı
   struct Simple_Structure pack() // boyut belirtilmemiş, 1 bayt sınırına hizalama aya
     {
      char              c; // sizeof(char)=1
      short             s; // sizeof(short)=2
      int               i; // sizeof(int)=4
      double            d; // sizeof(double)=8
     };
//--- basit bir yapı örneği bildir
   Simple_Structure s;
```

Yapı alanları şu durumlara göre birbiri ardına hafızaya yerleştirilmelidir: bildiri sırası ve tip boyutu. Yapı boyutu 15'tir, dizilerdeki yapı alanlarına olan öteleme tanımsızdır.

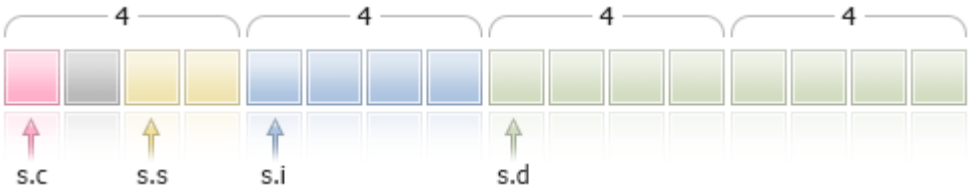


Şimdi aynı yapıyı 4 bayt olarak hizalayarak bildirin ve kodu çalıştırın.

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 4 baytlık hizalama ile basit yapı
   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
     };
   //--- basit bir yapı örneği bildir
   Simple_Structure s;
//--- her yapı üyesinin boyutunu göster
   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));
//--- şimdi, POD yapısının boyutunun, üyelerinin boyutlarının toplamına eşit olmadığı
   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 // yapı boyutu değişti
*/
  }
```

Yapı boyutu değişti, böylece 4 bayt ve daha fazla olan tüm üyeler, yapının başlangıcından itibaren 4 baytın katları kadar olacak şekilde ötelemeye sahip oldu. Daha küçük üyeler kendi boyut sınırlarına göre hizalanmalıdır (örneğin, 'short' için 2). Görünüşü şu şekildedir (eklenen bayt gri renkte gösterilir).



Bu durumda, **s.c** üyesinden sonra 1 bayt eklenmiştir, böylece **s.s** (sizeof(short)==2) alanı 2 bayt sınırına sahip olmuştur ('short' tipi için hizalama).

Dizideki yapının başlangıcına olan öteleme da 4-bayt sınırına hizalanır, diğer bir deyişle Simple_Structure arr[] için a[0], a[1] ve a[n] elemanlarının adresleri 4 baytın katları olacaktır.

4 baytlık hizalamaya ancak farklı üye sırasına sahip benzer tiplerden oluşan iki yapı daha ele alalım. İlk yapıda, üyeler artan tip boyutu sırasında bulunurlar.

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 4 bayt sınırına göre düzenlenmiş basit yapı
   struct CharShortInt pack(4)
     {
      char              c; // sizeof(char)=1
```

```
      short             s; // sizeof(short)=2
      int               i; // sizeof(double)=4
     };
//--- basit bir yapı örneği bildir
   CharShortInt ch_sh_in;
//--- her yapı üyesinin boyutunu göster
   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));

//--- POD yapısının büyüklüğünün, üyelerinin büyüklüğünün toplamına eşit olduğundan er
   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
*/
  }
```

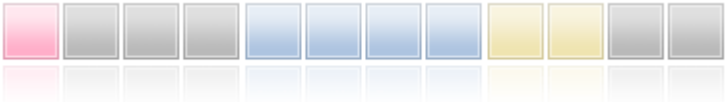
Gördüğümüz üzere, yapı boyutu 8'dir ve iki adet 4 baytlık bloktan oluşur. İlk blok 'char' ve 'short' tipindeki alanları, ikincisi ise 'int' tipindeki alanı içerir.

Şimdi, ilk yapıyı, 'short' tipi üyesinin sona taşınmasıyla sadece alan sırasında farklılık gösteren ikinci yapıya dönüştürelim.

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 4 bayt sınırına göre düzenlenmiş basit yapı
   struct CharIntShort pack(4)
     {
      char              c; // sizeof(char)=1
      int               i; // sizeof(double)=4
      short             s; // sizeof(short)=2
     };
//--- basit bir yapı örneği bildir
   CharIntShort ch_in_sh;
//--- her yapı üyesinin boyutunu göster
   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));
//--- POD yapısının büyüklüğünün, üyelerinin büyüklüğünün toplamına eşit olduğundan er
```

```
   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
*/
  }
```

Yapı içeriği değişmemiş olsa da, üye sırasını değiştirmek yapının boyutunu arttırdı.



Devralınırken hizalama da göz önünde bulundurulmalıdır. Bunu tek 'char' tipi üyesine sahip basit Parent yapısını kullanarak gösterelim. Hizasız yapı boyutu 1'dir.

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

'short' (sizeof(short)=2) tipi üyesini içeren Children alt sınıfını oluşturalım.

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

Sonuç olarak, 2 bayta hizalama yapılırken, üyelerinin boyutu 3 olmasına rağmen yapı boyutu 4'e eşittir. Bu örnekte, Parent sınıfına 2 bayt ayrılacaktır, böylece alt sınıfın 'short' alanına erişim 2 bayt ile hizalanmış olacaktır.

Bir MQL5 uygulaması, dosyalar veya akışlar düzeyinde yazarak/okuyarak üçüncü taraf verileriyle etkileşime girerse, yapı üyeleri için belleğin nasıl ayrıldığının bilgisi gereklidir.

Standart Kütüphanenin MQL5\Include\WinAPI dizini WinAPI fonksiyonlarıyla çalışmak için fonksiyonlar içerir. Bu fonksiyonlar, WinAPI ile çalışmak için gereken durumlarda, belirtilen bir hizalama ile yapıları uygular.

offsetof doğrudan pack özniteliği ile ilgili özel bir komuttur. Yapının başından bir üye ötelemesi elde etmemizi sağlar.

```
//--- Children tipi değişkeni bildir
   Children child;
//--- yapının başlangıcından ötelemeleri (ofsetleri) sapta
   Print("offsetof(Children,c)=",offsetof(Children,c));
   Print("offsetof(Children,s)=",offsetof(Children,s));
/*
  Result:
```

```
   offsetof(Children,c)=0
   offsetof(Children,s)=2
*/
```

'final' Şekillendiricisi

Yapıların bildirimi sırasında 'final' şekillendiricisinin kullanımı, kalıtımı engeller. Yapı üzerinde daha fazla değişikliğe ihtiyaç duyulmuyorsa veya güvenlik gerekçeleriyle izin verilmiyoyarsa, yapıyı bildirirken 'final' şekillendiricisini kullanabilirsiniz. Bu şekilde yapının tüm elemanlarının -kapalı yolla- son şekillerini aldığı varsayılır.

```
struct settings final
  {
  //--- Yapı gövdesi
  };

struct trade_settings : public settings
  {
  //--- yapı gövdesi
  };
```

Yukarıdaki örnekte de gösterildiği gibi, 'final' şekillendiricisi ile bildirilmiş bir yapıdan kalıtım yapılmak istendiğinde, derleyici hata dönüşü yapacaktır:

```
cannot inherit from 'settings' as it has been declared as 'final'
see declaration of 'settings'
```

Sınıflar

Sınıflar aşağıda belirtilen maddeler açısından yapılardan ayrılır:

- class anahtar sözcüğü bildirimde kullanılır;
- Aksi belirtilmemişse, varsayılan olarak tüm sınıf üyeleri private erişim belirtecine sahiptir. Aksi belirtilmemişse, yapıların veri üyeleri 'public' erişim belirtecine sahiptir;
- Sınıf içinde bildirilen bir sanal fonksiyon olmasa bile sınıf üyeleri her zaman bir sanal fonksiyonlar tablosuna sahiptir. Yapılar sanal fonksiyonlara sahip olamaz;
- new operatörü sınıf nesnelerine uygulanabilir ama yapılara uygulanamaz;
- sınıflar, diğer sınıflardan kalıtsal yolla oluşturulabilir. Aynı şekilde yapılar da sadece yapılardan kalıtsal olarak oluşturulabilir.

Sınıflar ve yapılar açık yapıcı ve yıkıcı fonksiyonlara sahip olabilirler. Eğer yapı fonksiyon açık şekilde tanımlanmışsa, yapı veya sınıf değişkeninin başlatma sırası kullanılarak başlatılması imkansızdır.

Örnek:

```
struct trade_settings
  {
   double take;         // kar sabitleme değerleri
   double stop;         // koruyucu durdurma fiyatı değeri
   uchar  slippage;     // kabul edilebilir slipaj
   //--- Yapıcı
          trade_settings() { take=0.0; stop=0.0; slippage=5; }
   //--- Yıkıcı
```

```
         ~trade_settings() { Print("Bu, son"); }
  };
//--- Derleyici başlatmanın imkansız olduğunu belirten bir hata mesajı oluşturacaktır
trade_settings my_set={0.0,0.0,5};
```

Yapıcı ve Yıkıcı Fonksiyonlar

Yapıcı, bir yapı veya sınıf nesnesi oluşturulduğunda otomatik olarak çağrılan özel bir fonksiyondur. Genellikle sınıf üyelerini başlatmak için kullanılır. Aynı uygulamalar yapılar için de geçerli olacağından, bundan sonra aksi belirtilmedikçe sadece sınıflar hakkında konuşacağız. Yapıcının ismi sınıf ismi ile örtüşmelidir. Yapıcının bir dönüş tipi yoktur (void tipini belirtebilirsiniz).

Tanımlanmış sınıf üyeleri - dizeler, dinamik diziler ve başlatma gerektiren nesneler - bir yapıcı olup olmadığına bakılmaksızın her durumda başlatılır.

Her sınıf parametre sayısı açısından veya başlatma listesi açısından farklı olan birden fazla yapıcı fonksiyon içerebilir. Parametrelerinin belirtilmesi gereken yapıcılara parametrik yapıcı denir.

Parametresiz olarak bildirilen yapıcılar **ön-tanımlı yapıcılardır**. Eğer sınıf içinde hiç yapıcı bildirilmemişse derleyici otomatik olarak bir ön-tanımlı yapıcı oluşturacaktır.

```
//+------------------------------------------------------------------+
//| Tarihle çalışmak için bir sınıf                                  |
//+------------------------------------------------------------------+
class MyDateClass
  {
private:
   int               m_year;           // Yıl
   int               m_month;          // Ay
   int               m_day;            // Ayın günü
   int               m_hour;           // Günün saati
   int               m_minute;         // Dakikalar
   int               m_second;         // Saniyeler
public:
   //--- Ön tanımlı yapıcı
                     MyDateClass(void);
   //--- Parametrik yapıcı
                     MyDateClass(int h,int m,int s);
  };
```

Yapıcı fonksiyonlar sınıf tarifinde bildirilebilir ve gövdeleri daha sonra tanımlanabilir. Örneğin MyDateClass'ın iki yapıcısı, şu yolla tanımlanabilir:

```
//+------------------------------------------------------------------+
//| Ön tanımlı yapıcı                                                |
//+------------------------------------------------------------------+
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__);
  }
//+------------------------------------------------------------------+
//| Parametrik yapıcı                                                |
//+------------------------------------------------------------------+
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__);
  }
```

Ön-tanımlı yapıcının içindeki tüm sınıf üyeleri TimeCurrent() fonksiyonu ile doldurulur. Parametrik yapıcı içindeyse sadece saat verileri kullanılır. Diğer sınıf üyeleri (m_year, m_month ve m_day) otomatik olarak mevcut tarihle başlatılır.

Ön tanımlı yapıcı kendi sınıfı içindeki bir nesne dizisi başlatılırken özel bir amaca sahiptir. Tüm parametreleri ön-tanımlı değerler alan yapılar ön-tanımlı yapı **değildir**. Bir örnek:

```
//+------------------------------------------------------------------+
//| Ön-tanımlı yapıcıya sahip bir sınıf                              |
//+------------------------------------------------------------------+
class CFoo
  {
   datetime          m_call_time;     // Son nesne çağrısının zamanı
public:
   //--- Ön tanımlı bir parametreye sahip olan yapıcı, bir ön tanımlı yapıcı değildir
                     CFoo(const datetime t=0){m_call_time=t;};
   //--- Kopya yapıcı
                     CFoo(const CFoo &foo){m_call_time=foo.m_call_time;};

   string ToString(){return(TimeToString(m_call_time,TIME_DATE|TIME_SECONDS));};
  };
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
```

```
void OnStart()
  {
// CFoo foo; // Bu varyant kullanılamaz - bir ön tanımlı yapıcı ayarlanmamış
//--- CFoo nesnesini oluşturmak için seçenekler
   CFoo foo1(TimeCurrent());     // Bir parametrik yapıcının açık çağrısı
   CFoo foo2();                  // Ön tanımlı parametreye sahip bir parametrik yapıcı
   CFoo foo3=D'2009.09.09';      // Bir parametrik yapıcının gizli çağrısı
   CFoo foo40(foo1);             // Kopya yapıcının açık çağrısı
   CFoo foo41=foo1;              // Kopya yapıcının kapalı çağrısı
   CFoo foo5;                    // Bir ön tanımlı yapıcının açık çağrısı ( eğer hiç
                                 // o zaman ön tanımlı değere sahip bir parametrik yap
//--- CFoo işaretçilerinin alınması için muhtemel seçenekler
   CFoo *pfoo6=new CFoo();       // Nesnenin dinamik olarak oluşturulması ve buna atan
   CFoo *pfoo7=new CFoo(TimeCurrent());// Dinamik nesne oluşturmak için bir diğer seçe
   CFoo *pfoo8=GetPointer(foo1); // şimdi pfoo8, foo1 nesnesini işaretliyor
   CFoo *pfoo9=pfoo7;            // pfoo9 ve pfoo7 aynı nesneye işaret ediyor
   // CFoo foo_array[3];         // Bu seçenek kullanılamaz - bir ön tanımlı yapıcı be
//--- m_call_time değerini göster
   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());
//--- Dinamik olarak oluşturulmuş dizileri sil
   delete pfoo6;
   delete pfoo7;
   //delete pfoo8;  // pfoo8, otomatik olarak foo1 oluşturulmuş nesnesini gösterdiğinc
   //delete pfoo9;  // pfoo7 ile aynı nesneye işaret ettiğinden pfoo9'u açıkça silmek
  }
```

Bu dizeleri yorumsuz bırakırsanız

```
   //CFoo foo_array[3];     // Bu varyant kullanılamaz - ön tanımlı bir yapıcı ayarlanm
```

veya

```
   //CFoo foo_dyn_array[];  // Bu varyant kullanılamaz - ön tanımlı bir yapıcı ayarlanm
```

Bu durumda derleyici, "ön tanımlı yapıcı ayarlanmamış" şeklinde bir hata verecektir.

Eğer bir sınıf kullanıcı tanımlı bir yapıcıya sahipse derleyici tarafından ön tanımlı bir yapıcı oluşturulmaz. Bu demektir ki, eğer sınıfın içinde bir ön tanımlı yapıcı değil, bir parametrik yapıcı bildirilmişse bu sınıfın nesnelerinin dizilerini bildiremezsiniz. Derleyici bu betik için bir hata dönüşü yapacaktır:

```
//+------------------------------------------------------------------+
//| Ön tanımlı yapıcısı olmayan bir sınıf                            |
```

```
//+------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
                     CFoo(string name) { m_name=name;}
  };
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Derleme sırasında "ön tanımlı yapıcı tanımlanmamış" hatasını al
   CFoo badFoo[5];
  }
```

Bu örnekte, CFoo sınıfı bir parametrik yapıcıya sahiptir - böyle durumlarda derleyici, derleme esnasında otomatik olarak bir ön tanımlı yapıcı oluşturmaz. Aynı zamanda, bir nesne dizisi bildirdiğinizde tüm nesnelerin otomatik olarak oluşturulması ve başlatılması gerektiği varsayılır. Bir nesnenin otomatik olarak başlatılması sırasında, bir ön tanımlı yapıcı çağrılması gerekir ama ön tanımlı yapıcı açıkça bildirilmedikçe veya derleyici tarafından otomatik olarak oluşturulmadıkça böyle bir nesneyi oluşturmak imkansızdır. Bu yüzden derleyici, derleme sırasında bir hata oluşturur.

Bir nesneyi, bir yapı aracılığıyla başlatmak için özel bir sözdizim mevcuttur. Bir sınıf veya yapı için, yapı başlatıcıları (başlatma için özel yapılar) başlatma listesinde belirlenebilir.

Bir başlatma listesi, yapıcının parametre listesi sonundaki iki noktanın ardından gelen, virgül ile ayrılmış bir başlatıcılar listesidir; gövdeden önce gelir (açılış parantezinin öncesinde). Birkaç gereksinimi vardır:

- Başlatma listeleri sadece yapıcılarda kullanılabilir;
- Ebeveyn üyeler başlatma listesinin içinde başlatılamaz;
- Başlatma listesi fonksiyon tanımından (uygulamasından) önce gelmelidir.

Burada, sınıf üyelerinin başlatılması için birkaç yapıcıya örnek verilmiştir.

```
//+------------------------------------------------------------------+
//| Bir karakterin ismini saklaması için bir sınıf                   |
//+------------------------------------------------------------------+
class CPerson
  {
   string            m_first_name;     // İlk isim
   string            m_second_name;    // Soy isim
public:
   //--- Boş bir ön tanımlı yapıcı
                     CPerson() {Print(__FUNCTION__);};
   //--- Bir parametrik yapıcı
                     CPerson(string full_name);
   //--- Başlatma listeli bir yapıcı
                     CPerson(string surname,string name): m_second_name(surname), m_fi
   void PrintName(){PrintFormat("İsim=%s Soyisim=%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);
     }
  }
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Bir hata al "ön tanımlı yapıcı tanımlanmamış"
   CPerson people[5];
   CPerson Tom="Tom Sawyer";                        // Tom Sawyer
   CPerson Huck("Huckleberry","Finn");              // Huckleberry Finn
   CPerson *Pooh = new CPerson("Winnie","Pooh");  // Winnie the Pooh
   //--- Çıktı değerleri
   Tom.PrintName();
   Huck.PrintName();
   Pooh.PrintName();

   //--- Dinamik olarak oluşturulmuş nesneleri sil
   delete Pooh;
  }
```

Bu durumda CPerson sınıfının üç yapıcısı vardır:

1. Bu sınıfın nesnelerinden bir dizi oluşturmayı sağlayan açık bir ön tanımlı yapıcı;
2. Bütün ismi bir parametre olarak alan ve sonra bunu, bulunan alanlara göre ilk isim ve soy isim olarak bölen tek parametreli bir yapıcı;
3. Bir başlatma listesi içeren iki parametreli bir yapıcı. Başlatıcılar - m_second_name(surname) and m_first_name(name).

Liste ile başlatma işleminin atamanın yerini değiştirdiğini not edin. Tekil üyeler şu şekilde başlatılmalıdır:

```
class_member (ifadelerden oluşan bir liste)
```

Başlatma listesinde üyeler herhangi şekilde sıralanabilir ama bütün sınıf üyeleri duyuruldukları sıra ile başlatılır. Bu demektir ki, üçüncü yapıcıda ilk olarak m_first_name üyesi başlatılır çünkü ilk o bildirilmiştir ve ancak ondan sonra m_second_name başlatılır. Bazı üyelerin başlatılmasının diğer sınıf üyelerinin değerlerine bağlı olduğu durumlarda, bu hesaba katılmalıdır.

Eğer temel sınıf içinde bir ön tanımlı yapıcı bildirilmişse ve parametreli olarak bir veya birden fazla yapıcı bildirilmişse, her zaman başlatma listesindeki temel sınıf yapıcılarından birini çağırmalısınız. Bunlar, listenin sıradan üyeleri gibi virgülle geçer ve başlatma listesinin nerede konumlandığına bakılmaksızın, nesnenin başlatılması sırasında ilk olarak çağrılır.

```
//+------------------------------------------------------------------+
//| Temel sınıf                                                      |
//+------------------------------------------------------------------+
class CFoo
  {
   string            m_name;
public:
   //--- Başlatma listeli bir yapıcı
                     CFoo(string name) : m_name(name) { Print(m_name);}
  };
//+------------------------------------------------------------------+
//| CFoo sınıfından türetilen sınıf                                  |
//+------------------------------------------------------------------+
class CBar : CFoo
  {
   CFoo              m_member;      // Bir sınıf üyesi ebeveynin nesnesidir
public:
   //--- Başlatma listesindeki bir ön tanımlı yapıcı ebeveyn yapıcısını çağırır
                     CBar(): m_member(_Symbol), CFoo("CBAR") {Print(__FUNCTION__);}
  };
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   CBar bar;
  }
```

Bu örnekte, bar nesnesini oluştururken bir ön tanımlı yapıcı CBar() çağrılacaktır. Bunun için önce ebeveyn CFoo için bir yapıcı çağrılır ve ardından m_member sınıf üyesi gelir.

Yıkıcı, bir sınıf nesnesi yok edildiğinde çağrılan özel bir fonksiyondur. Yıkıcının ismi bir sınıf ismi gibi tilde işareti ile (~) yazılır. Sonlandırma gerektiren dizgiler, dinamik diziler ve nesneler, yıkıcının varlığından bağımsız olarak sonlandırılır. Eğer bir yıkıcı varsa, o zaman bu eylemler yıkıcı çağrıldıktan sonra gerçekleştirilecektir.

Yıkıcılar, virtual anahtar sözcüğü ile bildirilmiş olup olmadıklarına bakılmaksızın daima sanaldır.

Sınıf Yöntemlerini Tanımlamak

Sınıf fonksiyon-yöntemleri sınıf içinde veya dışında tanımlanabilir. Eğer yöntem sınıf içinde tanımlanmışsa, gövde hemen yöntemin ardından gelir.

Örnek:

```
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[]);
  };
```

SetRightBorder(int border)'dan, Draw()'a kadar fonksiyonlar, doğrudan CTetrisShape sınıfının içinde tanımlanırlar.

CTetrisShape() yapıcısı ve CheckDown(int& pad_array[]), CheckLeft(int& side_row[]) ve CheckRight(int& side_row[]) yöntemleri sadece sınıf içinde tanımlanırlar ama henüz tanımlanmamış durumdadırlar. Bu fonksiyonların tanımları daha ilerde kodda yer alacaktır. Yöntemi sınıf dışında tanımlamak için kapsam çözünürlük operatörü kullanılır (sınıf ismi kapsam olarak kullanılır).

Örnek:

```
//+------------------------------------------------------------------+
//| Temel sınıfın yapıcısı                                           |
//+------------------------------------------------------------------+
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;
  }
//+------------------------------------------------------------------+
//| Aşağı hareket kabiliyetinin kontrol edilmesi (çubuk ve küp için) |
//+------------------------------------------------------------------+
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);
  }
```

Public, Protected ve Private Erişim Şekillendiricileri

Yeni bir sınıf geliştirirken üyelere dışarıdan erişimin sınırlanması tavsiye edilir. Bunun için private veya protected anahtar sözcükleri kullanılır. İlk durumda gizli verilere sadece aynı sınıfın içindeki yöntemler erişebilir. Eğer *protected* anahtar sözcüğü kullanılmışsa, gizli verilere kalıtımsal türetik sınıfların yöntemleri de erişebilecektir. Aynı yöntem sınıf yöntemlerine yapılacak erişimi engellemek için de kullanılabilir.

Bir sınıfın üyelerine ve/veya yöntemlerine erişimi tamamen açmanız gerekiyorsa public anahtar sözcüğünü kullanın.

Örnek:

```
class CTetrisField
  {
private:
   int               m_score;                           // Skor
   int               m_ypos;                            // Şekillerin mevcut konumla
   int               m_field[FIELD_HEIGHT][FIELD_WIDTH]; // Kuyu matrisi
   int               m_rows[FIELD_HEIGHT];              // Kuyu satırlarının numarala
   int               m_last_row;                        // Son boş satır
   CTetrisShape     *m_shape;                           // Tetris şekli
   bool              m_bover;                           // Oyun bitti
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();
  };
```

public: anahtar sözcüğünden sonra (ve bir sonraki erişim belirtecinden önce) bildirilen her sınıf üyesi ve yöntemi, programın sınıfa yapacağı her referansta kullanılabilir olacaktır. Bunlar örnek içinde şu üyelere karşılık gelir: CTetrisField(), Init(), Deinit(), Down(), Left(), Right(), Rotate() ve Drop() fonksiyonları.

private: anahtar sözcüğünden sonra (ve bir sonraki erişim belirtecinden önce) bildirilen her sınıf üyesi ve yöntemi, sadece bu sınıfın üyeleri-fonksiyonları için kullanılabilir. Erişim belirteçleri daima iki nota (:) ile sonlandırılır ve sınıf tanımında defalarca görülebilir.

protected: erişim belirteci sonrasında bildirilen tüm sınıf üyeleri (bir sonraki erişim belirtecine kadar), yalnızca bu sınıfın üye fonksiyonları ve bu sınıfın nesillerinin üye fonksiyonları için kullanılabilir. Dışarıdan private ve protected belirteçleri olan üyelere erişmeye çalışırken derleme aşamasında hata alınacaktır. Örnek:

```
class A
  {
protected:
   //--- kopya operatörü yalnızca A sınıfı ve nesillerinde bulunur
   void operator=(const A &)
     {
     }
  };
class B
  {
   //--- A sınıfı nesnesi bildirildi
   A                 a;
  };
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- iki B tipi değişken bildir
   B b1, b2;
   //--- bir nesneyi diğerine kopyalamaya çalış
   b2=b1;
  }
```

Bu kodu derlerken, bir hata mesajı alınacaktır - uzak kopya operatörü çağrısı:

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

Aşağıdaki ikinci dizge daha ayrıntılı bir açıklama sağlar - B sınıfındaki kopya operatörü, A sınıfının kullanılamayan kopya operatörü çağrıldığından açıkça silinmiştir:

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

Temel sınıfın üyelerine yapılacak erişim, türetilmiş sınıflardaki kalıtım sırasında yeniden tanımlanabilir.

'delete' belirteci

delete belirteci, kullanılamayan sınıf üye fonksiyonlarını işaretler. Bu, eğer program açıkça veya dolaylı olarak böyle bir fonksiyona atıfta bulunuyorsa, derleme aşamasında çoktan hata alınacağı anlamına gelir. Örneğin, bu belirteç, bir alt sınıfta üst yöntemlerin kullanılamamasını sağlar. Fonksiyonu, eğer üst sınıfın özel alanında bildirirsek (private bölümündeki bildirimler), aynı sonuç elde edilebilir. Burada, delete kullanmak, kodu nesiller seviyesinde daha okunabilir ve yönetilebilir hale getirir.

```
class A
  {
public:
                     A(void) {value=5;};
   double            GetValue(void) {return(value);}
private:
   double            value;
  };
class B: public A
  {
   double            GetValue(void)=delete;
  };
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- A tipi değişkeni bildir
   A a;
   Print("a.GetValue()=", a.GetValue());
//--- B tipi değişkenden değer elde et
   B b;
   Print("b.GetValue()=", b.GetValue()); // derleyici bu dizgede bir hata göstermekte
  }
```

Derleyici mesajı:

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

'delete' belirteci, otomatik dönüşümün veya kopya yapıcısının devre dışı bırakılmasına olanak tanır (aksi takdirde private bölümünde de gizlenmesi gerekirdi). Örnek:

```
class A
  {
public:
   void              SetValue(double v) {value=v;}
   //--- int tipi cağrıyı devre dışı bırak
   void              SetValue(int) = delete;
   //--- kopya operatörünü devre dışı bırak
```

```
   void              operator=(const A&) = delete;
private:
   double            value;
  };
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- iki A tipi değişken bildir
   A a1, a2;
   a1.SetValue(3);      // hata!
   a1.SetValue(3.14);   // TAMAM
   a2=a1;               // hata!
  }
```

Derleme sırasında şu hata mesajlarını almaktayız:

```
attempting to reference deleted function 'void A::SetValue(int)'
   function '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
```

'final' Şekillendiricisi

Sınıfın bildirimi sırasında 'final' şekillendiricisinin kullanımı, kalıtımı engeller. Sınıf üzerinde daha fazla değişikliğe ihtiyaç duyulmuyorsa veya güvenlik gerekçeleriyle izin verilmiyorsa, sınıf bildiriminde 'final' şekillendiricisini kullanın. Bu şekilde sınıfın tüm elemanlarının -kapalı yolla- son şekillerini aldığı varsayılır.

```
class CFoo final
  {
  //--- Sınıf gövdesi
  };

class CBar : public CFoo
  {
  //--- Sınıf gövdesi
  };
```

Yukarıdaki örnekte de gösterildiği gibi, 'final' şekillendiricisi ile bildirilmiş bir sınıftan kalıtım yapılmak istendiğinde derleyici hata dönüşü yapacaktır:

```
cannot inherit from 'CFoo' as it has been declared as 'final'
see declaration of 'CFoo'
```

Birleşimler (union)

Birleşim, aynı bellek alanını paylaşan birkaç değişkenden oluşan özel bir veri türüdür. Bu nedenle, birleşim aynı bit dizisini iki (veya daha fazla) farklı yolla yorumlama yeteneği sağlar. Birleşim bildirimi, yapı bildirimine benzer ve union anahtar kelimesiyle başlar.

```
union LongDouble
```

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

Yapının aksine, çeşitli birleşim üyeleri aynı hafıza alanına aittir. Bu örnekte, LongDouble birleşimi, aynı bellek alanını paylaşan long ve double tipleriyle bildirilmektedir. long_value ve double_value değişkenleri çakıştığından (bellekte), birleşim deposunu aynı anda bir *long* tamsayı değeri ve bir *double* gerçek değeri yapmanın mümkün olmadığını unutmayın (bir yapıdan farklı olarak). Diğer taraftan, bir MQL5 programı, birleşimde bulunan verileri herhangi bir zamanda bir tamsayı (long) veya gerçek (double) değer olarak işleyebilir. Bu nedenle, birleşim aynı veri sırasını temsil etmek için iki (veya daha fazla) seçenek almaya izin verir.

Birleşim bildirimi sırasında, derleyici otomatik olarak değişken birleşimi içinde en büyük tipi (hacme göre) depolamak için yeterli bellek alanını ayırır. Aynı sözdizimi, birleşim elemanına erişim için yapılarda olduğu gibi kullanılır - nokta operatörü.

```
union LongDouble
{
  long   long_value;
  double double_value;
};
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   LongDouble lb;
//--- geçersiz -nan(ind) numarasını alın ve görüntüleyin
   lb.double_value=MathArcsin(2.0);
   printf("1.  double=%f                integer=%I64X",lb.double_value,lb.long_value);
//--- en büyük normalleştirilmiş değer (DBL_MAX)
   lb.long_value=0x7FEFFFFFFFFFFFFF;
   printf("2.  double=%.16e  integer=%I64X",lb.double_value,lb.long_value);
//--- en küçük pozitif normalleştirilmiş değer (DBL_MIN)
   lb.long_value=0x0010000000000000;
   printf("3.  double=%.16e  integer=%.16I64X",lb.double_value,lb.long_value);
  }
/*  Execution result
    1.  double=-nan(ind)                integer=FFF8000000000000
    2.  double=1.7976931348623157e+308  integer=7FEFFFFFFFFFFFFF
    3.  double=2.2250738585072014e-308  integer=0010000000000000
*/
```

Birleşimler, programın aynı bellek verilerini farklı şekillerde yorumlamasına izin verdiğinden, genellikle alışılmadık bir tip dönüşümü gerektiğinde kullanılır.

Birleşimler kalıtım'da yer alamazlar ve aynı zamanda doğası gereği statik üyelere sahip olamazlar. Diğer tüm yönleriyle, *union*, tüm üyeleri sıfır ötelemeye (ofsete) sahip bir yapı gibi davranır. Aşağıdaki tipler birleşim üyesi olamazlar:

- dinamik diziler
- stringler
- nesne işaretçileri ve fonksiyonlar
- sınıf nesneleri
- yapıcılar veya yıkıcılara sahip olan yapı nesneleri
- yukarıdaki 1-5 noktalar arasından üyelere sahip olan yapı nesneleri

Sınıflara benzer şekilde birleşim, yöntemlerin yanı sıra yapıcı ve yıkıcılara da sahip olabilir. Varsayılan olarak, birleşim üyeleri public erişim tipindedir. Özel elemanlar oluşturmak için private anahtar kelimesini kullanın. Tüm bu olasılıklar, color tipindeki bir rengin ARGB'ye nasıl dönüştürüleceğini (ColorToARGB() fonksiyonunun yaptığı gibi) gösteren örnekte sunulmaktadır.

```
//+------------------------------------------------------------------+
//| color(BGR)'ın ARGB'ye dönüşümü için birleşim                     |
//+------------------------------------------------------------------+
union ARGB
  {
   uchar             argb[4];
   color             clr;
   //--- yapıcılar
                     ARGB(color col,uchar a=0){Color(col,a);};
                    ~ARGB(){};
   //--- genele açık yöntemler
public:
   uchar   Alpha(){return(argb[3]);};
   void    Alpha(const uchar alpha){argb[3]=alpha;};
   color   Color(){ return(color(clr));};
   //--- Özel yöntemler
private:
   //+------------------------------------------------------------------+
   //| alfa kanalı değerini ve rengini ayarla                           |
   //+------------------------------------------------------------------+
   void    Color(color col,uchar alpha)
     {
      //--- clr üyesinin rengini ayarla
      clr=col;
      //--- Alfa bileşen değerini ayarla - opaklık seviyesi
      argb[3]=alpha;
      //--- R ve B bileşenlerinin baytlarını değiş tokuş et (Kırmızı ve Mavi)
      uchar t=argb[0];argb[0]=argb[2];argb[2]=t;
     };
  };
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 0x55, 55/255=21.6% anlamına gelir (0%, tamamen saydamdır)
```

```
   uchar alpha=0x55;
//--- color tipi 0x00BBGGRR olarak temsil edilir
   color test_color=clrDarkOrange;
//--- ARGB birleşiminden gelen bayt değerleri burada kabul edilir
   uchar argb[];
   PrintFormat("0x%.8X - %s, BGR=(%s) için 'color' tipinin nasıl göründüğü belirtilmek
               test_color,ColorToString(test_color,true),ColorToString(test_color));
//--- ARGB tipi 0x00RRGGBB olarak temsil edilir, RR ve BB bileşenleri yer değiştirmiş
   ARGB argb_color(test_color);
//--- bayt dizisini kopyala
   ArrayCopy(argb,argb_color.argb);
//--- ARGB temsilinde görünümün nasıl olduğu
   PrintFormat("0x%.8X - alfa kanalı=0x%.2x, ARGB=(%d,%d,%d,%d) ile ARGB temsili",
               argb_color.clr,argb_color.Alpha(),argb[3],argb[2],argb[1],argb[0]);
//--- opaklık düzeyini ekle
   argb_color.Alpha(alpha);
//--- ARGB'yi 'color' tipi olarak tanımlamayı dene
   Print("color olarak ARGB=(",argb_color.clr,")  alfa kanalı=",argb_color.Alpha());
//--- bayt dizisini kopyala
   ArrayCopy(argb,argb_color.argb);
//--- ARGB temsilinde görünümün nasıl olduğu
   PrintFormat("0x%.8X - alfa kanalı=0x%.2x, ARGB=(%d,%d,%d,%d) ile ARGB temsili",
               argb_color.clr,argb_color.Alpha(),argb[3],argb[2],argb[1],argb[0]);
//--- ColorToARGB() fonksiyonu sonuçları ile kontrol edin
   PrintFormat("0x%.8X - ColorToARGB(%s,0x%.2x) nin sonucu",ColorToARGB(test_color,alp
               ColorToString(test_color,true),alpha);
  }
/* Gerçekleşim sonucu
   0x00008CFF - clrDarkOrange için 'color' tipinin nasıl göründüğü belirtilmektedir, B
  0x00FF8C00 - alfa kanalı=0x00, ARGB=(0,255,140,0) ile ARGB temsili
   color olarak ARGB=(0,140,255)  alfa kanalı=85
   0x55FF8C00 - alfa kanalı=0x55, ARGB=(85,255,140,0) ile ARGB temsili
   0x55FF8C00 - ColorToARGB(clrDarkOrange,0x55)'nin sonucu
*/
```

Arayüzler

Arayüzler sınıflar tarafından daha sonra kullanılmak üzere belirli bir işlevselliği tanımlamaya yarar. Aslında bunlar üyeleri olmayan sınıflardır yapıcı ve yıkıcı fonksiyonları içeremezler. Arayüzlerin bünyesinde bildirimi yapılan tüm yöntemler -açık tanıma sahip olmasalar bile- sanal yöntemlerdir.

Arayüzler "interface" anahtar sözcüğü ile tanımlanırlar. Örnek:

```
//--- Hayvanları açıklamak için temel arayüz
interface IAnimal
  {
//--- arayüzün yöntemleri varsayılan olarak genel erişime açık
   void Sound();  // hayvanın çıkardığı ses
  };
```

```
//+------------------------------------------------------------------+
//|  CCat sınıfı IAnimal arayüzünden türetilmiştir                   |
//+------------------------------------------------------------------+
class CCat : public IAnimal
  {
public:
                     CCat() { Print("Kedi doğdu"); }
                    ~CCat() { Print("Kedi öldü");  }
   //--- IAnimal arayüzündeki Sound yönteminin uygulanması
   void Sound(){ Print("miyav"); }
  };
//+------------------------------------------------------------------+
//|  CDog sınıfı IAnimal arayüzünden türetilmiştir                   |
//+------------------------------------------------------------------+
class CDog : public IAnimal
  {
public:
                     CDog() { Print("Köpek doğdu"); }
                    ~CDog() { Print("Köpek öldü");  }
   //--- IAnimal arayüzündeki Sound yönteminin uygulanması
   void Sound(){ Print("hav"); }
  };
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- IAnimal tipi nesneler için bir işaretçi dizisi
   IAnimal *animals[2];
//--- IAnimal 'dan çocuk sınıfların oluşturulması ve bunların işaretçilerinin bir diz
   animals[0]=new CCat;
   animals[1]=new CDog;
//--- Her çocuk sınıf için IAnimal arayüzünün Sound() yöntemini çağır
   for(int i=0;i<ArraySize(animals);++i)
      animals[i].Sound();
//--- Nesnelerin silinmesi
   for(int i=0;i<ArraySize(animals);++i)
      delete animals[i];
//--- Çalıştırma sonuçları
/*
  Kedi doğdu
  Köpek doğdu
  miyav
  hav
  Kedi öldü
  Köpek öldü
*/
  }
```

Soyut sınıflar gibi arayüzler de kalıtım olmadan kullanılamazlar. Sadece diğer arayüzlerden türetilebilirler ve ilgili sınıflara ebeveynlik yapabilirler. Arayüzler her zaman genel görünürlüğe sahiptir.

Bildirimleri sınıfların veya yapıların bildirimlerine dahil edilemez ama arayüzün bir işaretçisi void * tipli bir değişkende saklanabilir. Genel olarak, sınıf nesnesi işaretçileri void * tipli değişkenlerde saklanabilir. void * işaretçisini belli bir sınıf nesnesinin işaretçisine dönüştürmek için dynamic_cast operatörünü kullanın. Dönüşüm mümkün olmadığında, dynamic_cast işlemi NULL ifadesi ile sonuçlanır.

Ayrıca bakınız

Nesne Yönelimli Programlama

Dinamik Dizi Nesnesi

Dinamik Diziler

Diziler en fazla 4 boyutlu olacak şekilde bildirilir. Dinamik dizileri (ilk köşeli parantezlerinin içinde değer belirtilmemiş diziler) bildirirken, derleyici otomatik olarak üstteki yapıdan bir değişken oluşturur (bir dinamik dizi nesnesi) ve düzgün başlatma için bir kod sağlar.

Dinamik dizinin bildirildiği bloğun kapsamı dışına çıkıldığında, diziye tahsis edilen bellek alanı otomatik olarak boşaltılır.

Örnek:

```
double matrix[][10][20]; // 3 boyutlu dinamik dizi
ArrayResize(matrix,5);   // ilk boyutun büyüklüğünü ayarla
```

Statik diziler

Bir dizinin tüm boyutları açıkça belirtildiğinde, derleyici gerekli bellek boyutunu önceden tahsis eder. Bu tip diziler statik olarak adlandırılır. Yine de derleyici, önceden tahsis edilmiş ara-bellekle ilişkili olan bir dinamik dizi nesnesi için ek bellek tahsis edebilir.

Dinamik dizi nesnesinin oluşturulması, statik diziyi parametre olarak bir fonksiyona geçirme ihtiyacından kaynaklanır.

Örnekler:

```
double stat_array[5]; // 1 boyutlu statik dizi
some_function(stat_array);
...
bool some_function(double& array[])
  {
   if(ArrayResize(array,100)<0) return(false);
  ...
   return(true);
  }
```

Yapılar İçinde Diziler

Bir statik dizi, bir yapının üyesi olarak bildirildiğinde dinamik dizi nesnesi oluşturulmaz. Bu, Windows API'de kullanılan veri yapıları ile uyumluluk oluşturmak için düşünülmüştür.

Ama yapı üyesi olarak bildirilen statik diziler MQL5 programlarına da geçirilebilirler. Bu durumda, parametre geçilirken statik dizi ile bağlantılı bir geçici dinamik dizi nesnesi (yapı üyesi) oluşturulur.

Ayrıca Bakınız

Dizi Fonksiyonlar, Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Oluşturulması ve Silinmesi

Matrisler ve vektörler

vector tipi, MQL5'te vektörlerle çalışılmasına olanak sağlayan özel bir veri tipidir. Bir vektör, double tipinde bir tek boyutlu dizidir. Fizik, geometri vb. dahil olmak üzere birçok bilim alanında kullanılan lineer cebirin temel kavramlarından biridir. Vektörler, lineer denklem sistemlerini çözmek için, 3D grafiklerde ve diğer uygulama alanlarında kullanılır. Vektörler toplanabilir ve çarpılabilir. Vektörler arasındaki uzunluk veya mesafe norm kavramı ile ifade edilir. Programlamada, vektörler genellikle, üzerinde sıradan vektör işlemlerinin olmayan, yani dizilerin toplanamadığı veya çarpılamadığı ve normları olmadığı, homojen eleman dizileriyle temsil edilir.

Vektörler, matrislerle çalışırken satır vektörleri ve sütun vektörleri olarak temsil edilebilir. Ayrıca lineer cebirdeki vektörler kovaryans ve kontravaryans kavramlarını kullanır. Bu kavramlar, bir MQL5 kodu yazarken herhangi bir fark yaratmaz, çünkü vektör tipindeki her bir nesnenin ne olduğuna yalnızca programcı karar verir. Örneğin, 3D grafikler söz konusu olduğunda bu, döndürme, yer değiştirme, sıkıştırma vb. vektörü olabilir.

Genel olarak, lineer cebir açısından, bir sayı aynı zamanda bir vektördür, ancak bu, tek boyutlu vektör uzayında geçerlidir. Vektörün kendisi, matrisin özel bir hali olarak düşünülebilir.

matrix tipi, MQL5'te matrisleri temsil eden başka bir özel veri tipidir. Bir matris aslında double tipinde iki boyutlu bir dizidir. Belirli türdeki veri kümeleriyle daha kolay çalışma için vektörler ve matrisler MQL5 diline eklenmiştir. Programcıların lineer cebirin tüm avantajlarından mümkün olan en basit ve matematiğe en çok benzeyen biçimde yararlanmalarını sağlarlar. Matrisler, matematikte lineer veya diferansiyel denklem sistemlerini kompakt şekilde yazmak için yaygın olarak kullanılır. Bu durumda matris satırlarının sayısı denklemlerin sayısına ve sütunların sayısı bilinmeyenlerin sayısına karşılık gelir. Sonuç olarak, lineer denklem sistemleri matris işlemleri ile çözülebilir.

Matrisler için aşağıdaki şu cebirsel işlemler tanımlıdır:

- Aynı boyutlu matrislerin eklenmesi;
- Uygun boyutlu matrislerin çarpımı: soldaki matrisin sütun sayısı, sağdaki matrisin satır sayısına eşit olmalıdır;
- Matris çarpım kuralına göre, bir matrisin bir sütun vektörü ile çarpımı ve bir satır vektörünün bir matrisle çarpımı. Burada vektör, matrisin özel bir halidir;
- Bir sayı ile, yani bir skaler ile matris çarpımı

Matematik birçok farklı matris türünü dikkate alır. Örneğin, birim, simetrik, ters simetrik, üst ve alt üçgensel matrisler ve diğer matrisler. Çeşitli normal formlar, matris teorisinde önemli bir rol oynar. Belirli dönüşümler yoluyla elde edilebilen, matrisin belirli bir kanonik formunu temsil ederler. Pratikte, örneğin stabilite gibi ek özelliklere sahip normal formlar kullanılır.

Vektörlerin ve matrislerin veya daha doğrusu bu veri tiplerinin özel yöntemlerinin kullanılması, matematiksel gösterime yakın daha basit, daha kısa ve daha anlaşılır kod oluşturulmasına olanak sağlar. Bu, programcıyı iç içe döngüler oluşturmaktan ve hesaplamaya dahil olan dizilerin doğru indekslenmesine sürekli dikkat etmekten kurtarır. Dolayısıyla bu yöntemlerin kullanılması karmaşık programların geliştirilmesinde güvenilirliği ve hızı artırır.

matrix ve vector tiplerinde uygulanan yöntemlerin listesi

matrix ve vector tipleri, ilgili NumPy kütüphanesi yöntemlerine karşılık gelen yöntemleri içerir. Bu yöntemleri kullanarak algoritmaları ve kodları Python'dan MQL5'e minimum çabayla çevirebilirsiniz. Birçok veri işleme görevi, matematiksel denklemler, sinir ağları ve makine öğrenimi görevleri, hazır Python yöntemleri ve kütüphaneleri kullanılarak çözülebilir.

matrix/vector yöntemi	NumPy'deki benzer yöntem	Açıklama
void matrix.Eye(const int rows, const int cols, const int ndiag=0)	eye	Köşegende 1'lerin ve diğer yerlerde 0'ların olduğu bir matris oluşturur
void matrix.Identity(const int rows)	identity	Ana köşegen üzerinde 1'lerin olduğu bir kare matris oluşturur
void matrix.Ones(const int rows, const int cols)	ones	1'lerle dolu ve istenilen satır ve sütun sayısına sahip yeni bir matris oluşturur
void matrix.Zeros(const int rows, const int cols)	zeros	0'larla dolu ve istenilen satır ve sütun sayısına sahip yeni bir matris oluşturur
void matrix.Full(const int rows, const int cols, const scalar value)	full	Skaler değerle dolu ve istenilen satır ve sütun sayısına sahip yeni bir matris oluşturur
void matrix.Copy(const matrix a)	copy	İlgili matrisin bir kopyasını oluşturur
void matrix.FromBuffer(const int rows, const int cols, const scalar array[], const int count=-1, const int offset=0)	frombuffer	Tek boyutlu bir diziden bir matris oluşturur
void matrix.FromFile(const int rows, condt int cols, const int file_handle, const int count=-1, const int offset=0)	fromfile	Bir metin veya ikili dosyadaki verilerden bir matris oluşturur
void vector.FromString(const string source, const string sep=" ")	fromstring	Bir dizgeden metin verilerine dayalı bir vektör oluşturur
void vector.Arange(const scalar start, const scalar stop, const scalar step=1)	arange	Belirli bir aralıkta eşit aralıklı değerlere sahip bir vektör oluşturur
void matrix.Diag(const vector v, const int ndiag=0)	diag	Bir köşegeni çıkarır veya bir köşegen matris oluşturur
void matrix.Tri(const int rows, const int cols, const int ndiag=0)	tri	İstenilen köşegende ve altında 1'lerin ve diğer yerlerde 0'ların olduğu bir matris oluşturur
void matrix.Tril(const int rows, const int cols, const scalar array[], const int ndiag=0)	tril	k'ıncı köşegenin üzerindeki elemanların 0'a çevrildiği matrisin bir kopyasını geri döndürür

matrix/vector yöntemi	NumPy'deki benzer yöntem	Açıklama
void matrix.Triu(const int rows, const int cols, const scalar array[], const int ndiag=0)	triu	k'ıncı köşegenin altındaki elemanların 0'a çevrildiği matrisin bir kopyasını geri döndürür
void matrix.Vander(const vector v, const int cols=-1, const bool increasing=false)	vander	Bir Vandermonde matrisi oluşturur
vector matrix.Row(const unsigned nrow)		Bir satır vektörü geri döndürür
vector matrix.Col(const unsigned ncol)		Bir sütun vektörü geri döndürür
unsigned matrix.Rows()		Matristeki satır sayısını geri döndürür
unsigned matrix.Cols()		Matristeki sütun sayısını geri döndürür
void matrix.Init()		Matrisi başlatır
matrix matrix.Transpose()	transpose	Matrisin eksenlerini tersine çevirir veya değiştirir; değiştirilmiş matrisi geri döndürür
matrix matrix.Dot(const matrix b)	dot	İki matrisin nokta çarpımı
matrix matrix.Inner(const matrix b)	inner	İki matrisin iç çarpımı
matrix matrix.Outer(const matrix b)	outer	İki matrisin dış çarpımı
matrix matrix.MatMul(const matrix b)	matmul	İki matrisin matris çarpımı
matrix matrix.MatrixPower(const int power)	matrix_power	Kare matrisi n kuvvetine (tam sayı) yükseltir
matrix matrix.Kron(const matrix b)	kron	İki matrisin Kronecker çarpımı
bool matrix.Cholesky(matrix& L)	cholesky	Matrisin Cholesky ayrışmasını hesaplar
bool matrix.QR(matrix& Q, matrix& R)	qr	Matrisin qr ayrışmasını hesaplar
bool matrix.SVD(matrix& U, matrix& V, vector& singular_values)	svd	Matrisin tekil değer ayrışmasını hesaplar
bool matrix.Eig(matrix& eigen_vectors, vector& eigen_values)	eig	Kare matrisin özdeğerlerini ve sağ özvektörlerini hesaplar
bool matrix.EigH(matrix& eigen_vectors, vector&	eigh	Hermit matrisinin özdeğerlerini ve özvektörlerini geri döndürür

matrix/vector yöntemi	NumPy'deki benzer yöntem	Açıklama
eigen_values)		
bool matrix.EigVals(vector& eigen_values)	eigvals	Genel matrisin özdeğerlerini hesaplar
bool matrix.EigValsH(vector& eigen_values)	eigvalsh	Hermit matrisinin özdeğerlerini hesaplar
bool matrix.LU(matrix& L, matrix& U)		Alt üçgensel matris ve üst üçgensel matrisin çarpımı olan matrisin LU ayrışması
bool matrix.LUP(matrix& L, matrix& U, matrix& P)		Yalnızca satır permütasyonları ile LU ayrışması olarak, kısmi pivotlu LUP ayrışması: PA=LU
double matrix.Norm(const norm)	norm	Matrisin veya vektörün normunu geri döndürür
double matrix.Cond(const norm)	cond	Matrisin koşul numarasını hesaplar
vector matrix.Spectrum()		AT*A çarpımının özdeğerlerinin kümesi olarak matrisin spektrumunu hesaplar
double matrix.Det()	det	Dizinin determinantını hesaplar
int matrix.Rank()	matrix_rank	Gauss yöntemini kullanarak dizinin matris sıralamasını geri döndürür
int matrix.SLogDet(int& sign)	slogdet	Dizinin determinantının işaretini ve logaritmasını hesaplar
double matrix.Trace()	trace	Matrisin köşegenlerinin toplamını geri döndürür
vector matrix.Solve(const vector b)	solve	Lineer matris denklemini veya lineer cebirsel denklem sistemini çözer
vector matrix.LstSq(const vector b)	lstsq	Lineer cebirsel denklem sisteminin en küçük kareler çözer (kare olmayan veya dejenere matrisler için)
matrix matrix.Inv()	inv	Matrisin (çarpımsal) tersini hesaplar
matrix matrix.PInv()	pinv	Moore-Penrose yöntemiyle matrisin yalancı tersini hesaplar
int matrix.Compare(const matrix matrix_c, const double epsilon) int matrix.Compare(const matrix matrix_c, const int digits)		Belirtilen hassasiyetle iki matrisin/vektörün öğelerini karşılaştırır

matrix/vector yöntemi	NumPy'deki benzer yöntem	Açıklama
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	Matris öğesinin iki yerine bir indeks aracılığıyla adreslenmesine olanak sağlar
double vector.ArgMax() double matrix.ArgMax() vector matrix.ArgMax(const int axis)	argmax	Maksimum değerin indeksini geri döndürür
double vector.ArgMin() double matrix.ArgMin() vector matrix.ArgMin(const int axis)	argmin	Minimum değerin indeksini geri döndürür
double vector.Max() double matrix.Max() vector matrix.Max(const int axis)	max	Matrisdeki/vektördeki maksimum değeri geri döndürür
double vector.Mean() double matrix.Mean() vector matrix.Mean(const int axis)	mean	Eleman değerlerinin aritmetik ortalamasını hesaplar
double vector.Min() double matrix.Min() vector matrix.Min(const int axis)	min	Matrisdeki/vektördeki minimum değeri geri döndürür
double vector.Sum() double matrix.Sum() vector matrix.Sum(const int axis)	sum	Belirtilen eksen(ler) boyunca da gerçekleştirilebilen matris/vektör öğelerinin toplamını geri döndürür
void vector.Clip(const double min_value,const double max_value) void matrix.Clip(const double min_value,const double max_value)	clip	Matrisin/vektörün öğelerini belirtilen geçerli değerler aralığıyla sınırlar
vector vector.CumProd() vector matrix.CumProd() matrix matrix.CumProd(const int axis)	cumprod	Belirtilen eksen boyunca olanlar da dahil olmak üzere matris/vektör öğelerinin kümülatif çarpımını geri döndürür
vector vector.CumSum() vector matrix.CumSum() matrix matrix.CumSum(const int axis)	cumsum	Belirtilen eksen boyunca olanlar da dahil olmak üzere matris/vektör öğelerinin kümülatif toplamını geri döndürür
double vector.Prod(const double initial=1)	prod	Belirtilen eksen boyunca da gerçekleştirilebilen matris/vektör

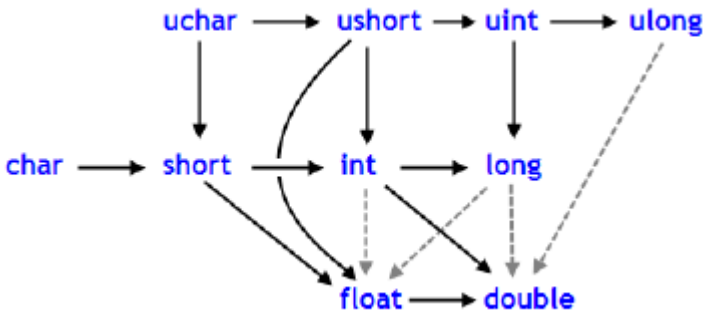
matrix/vector yöntemi	NumPy'deki benzer yöntem	Açıklama
double matrix.Prod(const double initial=1) vector matrix.Prod(const int axis,const double initial=1)		öğelerinin çarpımını geri döndürür
void matrix.Reshape(const ulong rows, const ulong cols)	reshape	Verilerini değiştirmeden matrisin şeklini değiştirir
void matrix.Resize(const ulong rows,const ulong cols)	resize	Belirtilen şekle sahip yeni bir matris geri döndürür
bool matrix.SwapRows(const ulong row1, const ulong row2)		Matristeki satırları değiştir
bool matrix.SwapCols(const ulong col1, const ulong col2)		Matristeki sütunları değiştir
double vector.Ptp() double matrix.Ptp() vector matrix.Ptp(const int axis)	ptp	Matrisin/vektörün veya belirtilen matris ekseninin Max() - Min()'e eşit olan değer aralığını geri döndürür
double vector.Percentile(const int percent) double matrix.Percentile(const int percent) vector matrix.Percentile(const int percent,const int axis)	percentile	Matris/vektör öğelerinin değerlerinin veya belirtilen eksen boyuncaki öğelerin değerlerinin belirtilen yüzdelik dilimini geri döndürür. 'percent' parametresinin geçerli değerleri [0, 100] aralığındadır
double vector.Quantile(const int percent) double matrix.Quantile(const int percent) vector matrix.Quantile(const int percent,const int axis)	quantile	Matris/vektör öğelerinin değerlerinin veya belirtilen eksen boyuncaki öğelerin değerlerinin belirtilen dağılım dilimini geri döndürür. 'percent' parametresi [0, 1] aralığında değerler alır
double vector.Median() double matrix.Median() vector matrix.Median(const int axis)	median	Matris/vektör öğelerinin medyanını hesaplar. Medyan, matris/vektör öğelerinin en yüksek yarısını, en düşük yarısından ayıran orta değerdir
double vector.Average() double matrix.Average() vector matrix.Average(const int axis)	average	Matris/vektör değerlerinin aritmetik ortalamasını hesaplar. Paydadaki ağırlıkların toplamı 0'a eşit olamaz, ancak bazı ağırlıklar 0 olabilir
double vector.Std() double matrix.Std() vector matrix.Std(const int axis)	std	Matris/vektör öğelerinin değerlerinin veya belirtilen eksen boyuncaki öğelerin değerlerinin standart sapmasını geri döndürür

matrix/vector yöntemi	NumPy'deki benzer yöntem	Açıklama
double vector.Var() double matrix.Var() vector matrix.Var(const int axis)	var	Matris/vektör öğelerinin değerlerinin varyansını hesaplar
double vector.CorrCoef(const vector& v) matrix matrix.CorrCoef()	corrcoef	Pearson korelasyon katsayısını (lineer korelasyon katsayısı) hesaplar. Korelasyon katsayısı [-1, 1] aralığındadır
vector vector.Correlate(const vector& v,enum mode)	correlate	İki vektörün çapraz korelasyonunu hesaplar. 'mode' parametresi lineer konvolüsyon hesaplama modunu belirler
vector vector.Convolve(const vector& v, enum mode)	convolve	İki vektörün ayrık, lineer konvolüsyonunu geri döndürür. 'mode' parametresi lineer konvolüsyon hesaplama modunu belirler
matrix matrix.Cov() matrix vector.Cov(const vector& v); (resulting matrix 2 x 2)	cov	Kovaryans matrisini hesaplar. İki örneğin (iki rastgele değişken) kovaryansı, onların lineer bağımlılığının bir ölçüsüdür

Tip Dönüşümü

Sayısal tipleri dönüştürmek

Sıklıkla, bir sayısal tipi diğerine dönüştürme ihtiyacı doğar. Tüm sayısal tipler birbirlerine dönüştürülemezler. İzin verilen dönüşümlerin şeması:



Oklu kalın çizgiler neredeyse hiç bilgi kaybı olmadan gerçekleştirilen değişimleri gösterir. Char tipinin yerine bool tipi kullanılabilir (ikisi de 1 bayt bellek kaplar), int tipinin yerine color tipi kullanılabilir (4 bayt), long tipinin yerine datetime kullanılabilir (8 bayt kaplarlar). Aralıklı çizgi oklar çözünürlük kaybı gerçekleşebilecek dönüşümleri gösterir. Örneğin bir tamsayıdaki (int) hanelerin sayısı 123456789'a eşittir . Bu, float tipiyle temsil edilebilecek hane sayısından büyüktür.

```
   int n=123456789;
   float f=n;     // içerik, 1.234567892E8'e eşittir
   Print("n = ",n,"   f = ",f);
   // sonuç n= 123456789    f= 123456792.00000
```

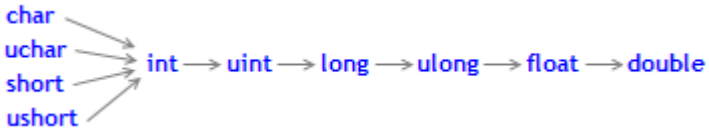
float tipine dönüştürülen bir sayı aynı düzende olur ama daha az doğrudur. Siyah okların ters yönünde yapılan dönüşümler, muhtemel veri kaybı ile gerçekleştirilebilir. char ve uchar, short ve ushort, int ve uint, long ve ulong dönüşümleri (iki yöne dönüşümler) veri kaybına yol açabilir.

Kayan nokta değerlerinin tamsayı tipine dönüşümünün bir sonucu olarak, ondalık kısım daima silinir. Eğer float tipli bir sayıyı en yakın tamsayıya yuvarlamak istiyorsanız (-ki bir çok durumda daha kullanışlıdır), MathRound() fonksiyonunu kullanmalısınız.

Örnek:

```
//--- Kütleçekim ivmesi
   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);
/*
  Sonuç:
   round_g = 9
   math_round_g = 10
*/
```

Eğer iki değer, bir ikili işlem ile birleştirilmişse; işlem gerçekleştirilmeden önce daha düşük tipli işlenen, altta yer alan şemadaki önceliğe göre daha yüksek olan tipe dönüştürülür:



Veri tipleri char, uchar, short ve ushort; istisnasız olarak int tipine dönüştürülürler.

Örnekler:

```
   char   c1=3;
//--- İlk örnek
   double d2=c1/2+0.3;
   Print("c1/2 + 0.3 = ",d2);
// Sonuç:   c1/2+0.3 = 1.3

//--- İkinci örnek
   d2=c1/2.0+0.3;
   Print("c1/2.0 + 0.3 = ",d2);
// Sonuç:   c1/2.0+0.3 = 1.8
```

Hesaplanan ifade iki işlemden oluşuyor. İlk örnekte, char tipli c1 değişkeni int tipi bir geçici değişkene dönüştürülür. Çünkü bölümdeki ikinci işlenen, sabit 2'nin tipi daha yüksektir. 3/2 tamsayı bölümünün sonucunda int tipli '1' değerini elde ederiz.

Birinci örneğin ikinci işleminde, ikinci işlenen double tipli '0.3' sabitidir. Bu yüzden, ilk işlemin sonucu 1.0 değeri ile double tipli geçici değişkene dönüştürülmüştür.

İkinci örnekte char tipli değişken c1, double tipli bir geçici değişkene dönüştürülmüştür, çünkü bölme işleminin ikinci işleneni sabit '2.0', double tiplidir. Başka dönüşüm yapılmamıştır.

Sayısal Tiplerin Dönüşümü

MQL5 dilindeki ifadelerde, açık ve gizli (kapalı) olmak üzere iki tip dönüşümü de kullanılabilir. Açık tip dönüşümü şu şekilde yazılır:

```
var_1 = (tip)var_2;
```

Bir ifade veya fonksiyon sonucu, var_2 değişkeni olarak kullanılabilir. Açık tip dönüşümünün fonksiyonel ifadesi de mümkündür:

```
var_1 = tip(var_2);
```

Birinci örnek temelinde bir açık tip dönüşümü düşünelim.

```
//--- Üçüncü örnek
   double d2=(double)c1/2+0.3;
   Print("(double)c1/2 + 0.3 = ",d2);
// Sonuç:   (double)c1/2+0.3 = 1.80000000
```

Bölme işlemi gerçekleştikten sonra c1 değişkeni açık olarak double tipine dönüştürülür. Ardından tamsayı sabiti 2, double tipli 2.0 değişkenine dönüşür. Dönüşümün sonucu olarak ilk işlenen, double tipini alır. Açık tip dönüşümü tekil bir işlemdir.

Ayrıca, tipleri dönüştürürken sonuç hoş görülebilir sınırların dışına çıkabilir. Bu durumda budama gerçekleşir. Örneğin:

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

İşlemler gerçekleştirilmeden önce (atama işlemleri hariç); veri, en yüksek öncelikli tipe dönüştürülür. Atama işlemlerinden önce ise veri, hedef tipe dönüştürülür.

Örnekler:

```
   int    i=1/2;         // tip dönüşümü yok, sonuç 0
   Print("i = 1/2  ",i);

   int k=1/2.0;          // ifade double tipine dönüştürüldü,
   Print("k = 1/2  ",k);  // sonra int hedef tipine dönüştürüldü, sonuç 0

   double d=1.0/2.0;     // tip dönüşümü yok, sonuç 0.5
   Print("d = 1/2.0; ",d);

   double e=1/2.0;       // ifade double tipine dönüştürüldü,
   Print("e = 1/2.0; ",e);// hedef tipi ile aynı, sonuç 0.5

   double x=1/2;         // ifade double hedef tipine dönüştürüldü,
   Print("x = 1/2; ",x);  // sonuç 0.0
```

long/ulong tipli değişkenler double tipine dönüştürüldüğünde, tamsayı değerinin 9223372036854774784'den büyük veya -9223372036854774784'den küçük olması durumunda çözünürlük kaybı olabilir.

```
void OnStart()
  {
   long l_max=LONG_MAX;
   long l_min=LONG_MIN+1;
//--- double tipine dönüştürürken çözünürlük kaybı olmayacak en yüksek tamsayı değeri
   while(l_max!=long((double)l_max))
      l_max--;
//--- double tipine dönüştürürken çözünürlük kaybı olmayacak en düşük tamsayı değerini
   while(l_min!=long((double)l_min))
      l_min++;
//--- elde edilen aralığı tamsayı değerlerle yaz
   PrintFormat("Bir tamsayı değerini double tipine dönüştürürken, "
               "değer, [%I64d, %I64d] aralığının içinde olmalı",l_min,l_max);
```

```
//--- şimdi, değer bu aralığın dışına düştüğünde ne oluyormuş ona bakalım
   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)));
//--- şu sonucu al
// Bir tamsayı değeri double tipine dönüştürüldüğünde, değer, [-9223372036854774784, 9
// 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_
  }
```

String Tipi için Tip Dönüşümü

string tipi basit tipler arasında en yüksek önceliğe sahiptir. Bu yüzden, işlenenlerden biri string tipinde ise, diğer işlenen de otomatik olarak string tipine dönüştürülecektir. Dizgilerde basit diyadik toplama işleminin mümkün olduğunu not ediniz. string tipinin açık tip dönüşümü, dönüştürülecek her sayısal tip için izinlidir.

Örnekler:

```
   string s1=1.0/8;            // ifade double tipine dönüştürüldü,
   Print("s1 = 1.0/8; ",s1);     //  sonra da string tipli hedefe,
// sonuç "0.12500000" (10 karakterli bir dizgi)

   string s2=NULL;             // dizginin sonlandırılması
   Print("s2 = NULL; ",s2);      // sonuç boş bir dizgi
   string s3="Ticket N"+12345; // ifade string tipine dönüştürüldü
   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));
```

Temel Sınıf İşaretçilerinden Türev Sınıf İşaretçilerine Tip Dönüşümü

Açık şekilde oluşturulmuş sınıf nesneleri, ilgili temel sınıfın nesneleri olarak da görülebilirler. Bu bazı ilginç sonuçlara yol açar. Örneğin, aynı temel sınıftan türetilmiş farklı sınıfların nesnelerinin, birbirlerinden belirgin şekilde farklı olabildikleri gerçeğini bir yana bırakırsak; bunları temel tipin nesneleri olarak gördüğümüz bir bağlantılı liste oluşturabiliriz. Ama tersi geçerli değildir: temel sınıf nesneleri otomatik olarak türetik sınıfların nesnesi olmazlar.

Temel sınıfın işaretçilerini, türetik sınıfın işaretçilerine dönüştürmek için açık dönüşüm kullanabilirsiniz. Fakat böyle bir dönüşümün kabul edilebilirliği konusunda emin olmalısınız. Çünkü, aksi durumda, önemli bir çalışma zamanı hatası gerçekleşir ve MQL5 programı durur.

dynamic_cast işlemcisi ile dinamik tip dönüşümü

Dinamik tip dönüşümü 'dynamic_cast' işlemcisiyle gerçekleştirilir ve sadece sınıf işaretçileri için kullanılabilir. Tipin doğrulaması çalıştırma sırasında gerçekleştirilir. Yani, dynamic_cast işlemcisi kullanıldığında, derleyici tip dönüşümüne konu olan veriyi kontrol etmeyecektir. İşaretçinin, nesnenin gerçek veri tipi dışındaki tiplere dönüştürülmesinin sonucunda NULL değeri alınır.

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

type-id parametresi daha önce tanımlanan bir sınıf tipini işaret etmelidir. C++ dilinin aksine, *expression* işlemci tipi, void haricinde herhangi bir değeri alabilir.

Örnek:

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

void OnStart()
  {
   CBar bar;
//--- *bar işaretçisini *foo işaretçisine dönüştürme iznimiz var
   CFoo *foo = dynamic_cast<CFoo *>(&bar); // kritik hata yok
   Print(foo);                             // foo=NULL
//--- Bar tipli bir nesne referansını Foo tipli bir nesneye açık yolla dönüştürme giri
   foo=(CFoo *)&bar;                       // kritik çalışma hatası
   Print(foo);                             // bu dizgi çalıştırılmaz
  }
```

Ayrıca Bakınız

Veri Tipleri

Void Tipi ve NULL Sabiti

Sözdizim açısından void tipi; char, uchar, bool, short, ushort, int, uint, color, long, ulong, datetime, float, double ve string tipleri ile birlikte temel tiplerden biridir. Bu tip, ya fonksiyonun herhangi bir değere dönüş yapmadığını belirtmek için, ya da parametre olarak fonksiyon parametrelerinin olmadığını göstermek için kullanılır.

Ön tanımlı sabit değişken **NULL** *void* tipindedir. Diğer temel tipteki değişkenlere dönüşüm olmaksızın atanabilir. Temel tip değişkenlerinin **NULL** değeri ile karşılaştırılmalarına izin verilir.

Örnek:

```
//--- Eğer dizgi başlatılmamışsa, ön tanımlı değeri buna ata
if(some_string==NULL) some_string="boş";
```

NULL aynı zamanda new operatörü ile oluşturulmuş nesne işaretçileri ile de karşılaştırılabilir.

Ayrıca Bakınız

Değişkenler, Fonksiyonlar

Kullanıcı tanımlı tipler

C++ da kullanılanılan typedef sözcüğü kullanıcı tanımlı veri tiplerinin oluşturulmasını sağlar. Bunun için varolan bir veri tipine yeni bir isim verin. Burada yeniş bir veri tipi oluşturulmayacaktır. Bunun yerine varolan tip için yeni bir isim tanımlanacaktır. Kullanıcı tanımlı tipler uygulamalarınızı daha esnek hale getirir: bazen ikame makroları (#define) kullanarak typedef'i değiştirmek yeterlidir. Kullanıcı tanımlı tipler kodun okunaklılığını artırır (çünkü typedef kullanımıyla standart veri tiplerine daha anlaşılır isimler verebilirsiniz). Kullanıcı tanımlı veri tipi oluşturmak için genel format:

```
   typedef type yeni_isim;
```

Burada, *type* kullanılabilir veri tipidir, *yeni_isim* ise söz konusu tipin yeni ismidir. Yeni isim sadece mevcut isme ek olarak ayarlanır (onun yerini almaz). MQL5 dili, typedef kullanımı ile fonksiyon işaretçileri oluşturabilmenizi sağlar.

Fonksiyon işaretçisi

Fonksiyon işaretçileri genelde şu şekilde tanımlanır

```
   typedef fonksiyon_sonuç_tipi (*Fonksiyon_isim_tipi)(giriş_parametrelerinin_tipleri)
```

burada typedef sözcüğünden sonra, fonksiyonun imzası (giriş parametrelerinin sayısı ve tipleri ve fonksiyonun dönüş tipi) ayarlanır. Aşağıda bir fonksiyona işaretçi ayarlanmasını gösteren bir örnek verilmiştir:

```
//--- int tipli iki giriş parametresi olan bir fonksiyonun işaretçisini bildir
   typedef int (*TFunc)(int,int);
//--- burada TFunc bir tiptir ve fonksiyon değişken işaretçisiyle bildirilebilir
   TFunc func_ptr; // fonksiyon işaretçisi
//--- TFunc tanımına karşılık gelen fonksiyonları bildir
   int sub(int x,int y) { return(x-y); }  // diğerinden bir sayı çıkar
   int add(int x,int y) { return(x+y); }  // iki sayının eklenmesi
   int neg(int x)       { return(~x);  }  // değişken bitlerini tersine çevir
//--- func_ptr değişkeni daha sonra kulanılmak üzere fonksiyonun adresini saklayabili
   func_ptr=sub;
   Print(func_ptr(10,5));
   func_ptr=add;
   Print(func_ptr(10,5));
   func_ptr=neg;           // hata: neg fonksiyonu int (int,int) tipine sahip değil
   Print(func_ptr(10));    // hata: iki parametre gerekli
```

Bu örnekte *func_ptr* değişkeni *sub* ve *add* fonksiyonlarını alabilir (int tipli iki girdi değişken *TFunc* fonksiyon işaretçisinde tanımlanmış). Aksine, *neg* fonksiyonu işareti farklı olduğu için func_ptr işaretçisine atanamaz.

Kullanıcı arayüzündeki olay modellerinin düzenlenmesi

Kullanıcı arayüzleri oluştururken fonksiyon işaretçileri ile olay işleyicileri oluşturebilirsiniz. Düğmeler oluşturmak ve bunlara basıldığında ilgili olayın işlenmesini sağlayacak fonksiyonları eklemek için CButton bölümünden bir örnek kullanalım. Önce, düğmeye basıldığında çağrılacak *TAction* fonksiyonunun işaretçisini tanımlayın ve *TAction açıklamasına göre üç fonksiyon oluşturun.*

```
//--- özel fonksiyon tipi oluştur
typedef int(*TAction)(string,int);
//+------------------------------------------------------------------+
//|  Dosyayı aç                                                      |
//+------------------------------------------------------------------+
int Open(string name,int id)
  {
   PrintFormat("%s fonksiyonu çağrıldı (isim=%s tanıtıcı=%d)",__FUNCTION__,name,id);
   return(1);
  }
//+------------------------------------------------------------------+
//|  Dosyayı kaydet                                                  |
//+------------------------------------------------------------------+
int Save(string name,int id)
  {
   PrintFormat("%s fonksiyonu çağrıldı (isim=%s tanıtıcı=%d)",__FUNCTION__,name,id);
   return(2);
  }
//+------------------------------------------------------------------+
//|  Dosyayı Kapat                                                   |
//+------------------------------------------------------------------+
int Close(string name,int id)
  {
   PrintFormat("%s fonksiyonu çağrıldı (isim=%s tanıtıcı=%d)",__FUNCTION__,name,id);
   return(3);
  }
```

Sonra, CButton sınıfını kullanarak MyButton sınıfını oluşturun, burada *TAction* işaretçisini fonksiyona eklemeliyiz.

```
//+------------------------------------------------------------------+
//| Olay işleme fonksiyonu olan bir düğme sınıfı oluştur             |
//+------------------------------------------------------------------+
class MyButton: public CButton
  {
private:
   TAction           m_action;                    // çizelge olayları işleyicisi
public:
                     MyButton(void){}
                    ~MyButton(void){}
   //--- olay işleme fonksiyonunun işaretçisini ve düğme metnini belirten yapıcı
                     MyButton(string text, TAction act)
     {
      Text(text);
      m_action=act;
     }
   //--- OnEvent() olay işleyicisinden çağrılacak özel fonksiyonu ayarla
   void              SetAction(TAction act){m_action=act;}
```

```
   //--- standart çizelge olayları işleyicisi
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
     {
      if(m_action!=NULL && lparam==Id())
        {
         //--- özel m_action() işleyicisini çağır
         m_action(sparam,(int)lparam);
         return(true);
        }
      else
      //--- CButton ana sınıfından çağrılan işleyicinin sonucunu göster
         return(CButton::OnEvent(id,lparam,dparam,sparam));
     }
  };
```

CControlsDialog sınıfını CAppDialog sınıfından türeterek oluştur, *MyButton* tipli düğmeleri kaydetmek için *m_buttons* dizisini ve *AddButton(MyButton &button)* ve *CreateButtons() yöntemlerini ekle*.

```
//+------------------------------------------------------------------+
//| CControlsDialog sınıfı                                           |
//| Amaç: uygulamanın yönetilmesi için kullanılacak grafik panel  |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CArrayObj         m_buttons;                    // düğme dizisi
public:
                     CControlsDialog(void){};
                    ~CControlsDialog(void){};
   //--- oluştur
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- düğmeyi ekle
   bool              AddButton(MyButton &button){return(m_buttons.Add(GetPointer(butto
protected:
   //--- düğmeleri oluştur
   bool              CreateButtons(void);
  };
//+------------------------------------------------------------------+
//| CControlsDialog nesnesini çizelgede oluştur                      |
//+------------------------------------------------------------------+
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());
//---
  }
//+------------------------------------------------------------------+
//| tanımlar                                                         |
```

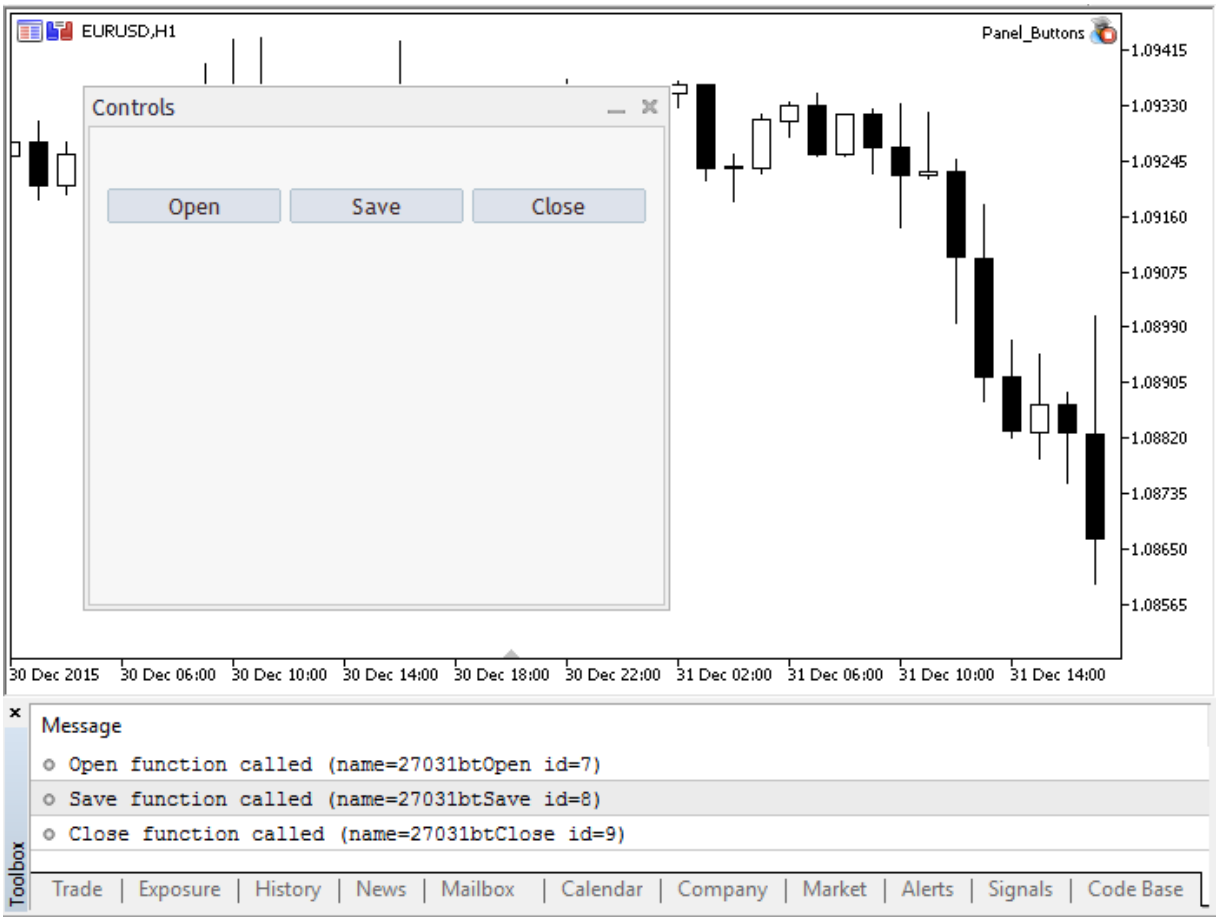
```
//+------------------------------------------------------------------+
//--- girintiler ve boşluklar
#define INDENT_LEFT                         (11)      // sol girinti (çerçeve payıyla
#define INDENT_TOP                          (11)      // üst girinti (çerçeve payıyla
#define CONTROLS_GAP_X                      (5)       // X koordinatına göre boşluk
#define CONTROLS_GAP_Y                      (5)       // Y koordinatına göre boşluk
//--- düğmeler için
#define BUTTON_WIDTH                        (100)     // X koordinatına göre genişkik
#define BUTTON_HEIGHT                       (20)      // Y koordinatına göre genişlik
//--- gösterge alanı için
#define EDIT_HEIGHT                         (20)      // Y koordinatına göre genişlik
//+------------------------------------------------------------------+
//| Düğmeleri oluşturup ControlsDialog paneline ekle                 |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButtons(void)
  {
//--- düğme koordinatlarını hesapla
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2;
   int y2=y1+BUTTON_HEIGHT;
//--- düğme nesnelerini fonksiyon işaretçilerine ekle
   AddButton(new MyButton("Open",Open));
   AddButton(new MyButton("Save",Save));
   AddButton(new MyButton("Close",Close));
//--- düğmeleri grafiksel olarak oluştur
   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("%s %d düğmesi oluşturulamadı",b.Text(),i);
         return(false);
        }
      //--- her düğmeyi CControlsDialog taşıyıcısına ekle
      if(!Add(b))
         return(false);
     }
//--- başarılı
   return(true);
  }
```

Şimdi, üç düğmeli (Open, Save ve Close) kontrol paneli CControlsDialog ile programı geliştirebiliriz. Bir düğmeye basıldığında, *TAction* işaretçisi biçimindeki uygun bir fonksiyon çağrılacaktır.

```
//--- program başlatıldığında otomatik olarak oluşturulması için nesneyi global seviye
CControlsDialog MyDialog;
```

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- şimdi nesneyi çizelgede oluştur
   if(!MyDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- uygulamayı çalıştır
   MyDialog.Run();
//--- uygulama başarıyla çalıştırıldı
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- iletişim kutusunu yok et
   MyDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // olay tanıtıcısı
                  const long& lparam,   // long tipli olay parametresi
                  const double& dparam, // double tipli olay parametresi
                  const string& sparam) // string tipli olay parametresi
  {
//--- çizelge olayları için gereken işleyiciyi (CAppDialog) ana sınıftan çağır
   MyDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Çalıştırılan uygulamanın görünümü ve düğmelerin tıklanmalarının sonuçları ekran görüntüsünde verilmiştir.



Programın tam kaynak kodu

```
//+------------------------------------------------------------------+
//|                                                Panel_Buttons.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+

#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Birkaç CButton düğmesi içeren panel"
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
//+------------------------------------------------------------------+
//| tanımlar                                                         |
//+------------------------------------------------------------------+
//--- girintiler ve boşluklar
#define INDENT_LEFT                         (11)      // sol girinti (çerçeve payıyla
#define INDENT_TOP                          (11)      // üst girinti (çerçeve payıyla
#define CONTROLS_GAP_X                      (5)       // X koordinatına göre boşluk
#define CONTROLS_GAP_Y                      (5)       // Y koordinatına göre boşluk
```

```
//--- düğmeler için
#define BUTTON_WIDTH                    (100)      // X koordinatına göre genişkik
#define BUTTON_HEIGHT                   (20)       // Y koordinatına göre genişlik
//--- gösterge alanı için
#define EDIT_HEIGHT                     (20)       // Y koordinatına göre genişlik

//--- özel fonksiyon tipini oluştur
typedef int(*TAction)(string,int);
//+------------------------------------------------------------------+
//|  Dosyayı aç                                                      |
//+------------------------------------------------------------------+
int Open(string name,int id)
  {
   PrintFormat("%s fonksiyonu çağrıldı (isim=%s tanıtıcı=%d)",__FUNCTION__,name,id);
   return(1);
  }
//+------------------------------------------------------------------+
//|  Dosyayı kaydet                                                  |
//+------------------------------------------------------------------+
int Save(string name,int id)
  {
   PrintFormat("%s fonksiyonu çağrıldı (isim=%s tanıtıcı=%d)",__FUNCTION__,name,id);
   return(2);
  }
//+------------------------------------------------------------------+
//|  Dosyayı Kapat                                                   |
//+------------------------------------------------------------------+
int Close(string name,int id)
  {
   PrintFormat("%s fonksiyonu çağrıldı (isim=%s tanıtıcı=%d)",__FUNCTION__,name,id);
   return(3);
  }
//+------------------------------------------------------------------+
//| Olay işleme fonksiyonu olan bir düğme sınıfı oluştur             |
//+------------------------------------------------------------------+
class MyButton: public CButton
  {
private:
   TAction           m_action;                    // çizelge olayları işleyicisi
public:
                     MyButton(void){}
                    ~MyButton(void){}
   //--- olay işleme fonksiyonunun işaretçisini ve düğme metnini belirten yapıcı
                     MyButton(string text,TAction act)
     {
      Text(text);
      m_action=act;
     }
   //--- OnEvent() olay işleyicisinden çağrılacak özel fonksiyonu ayarla
```

```
   void              SetAction(TAction act){m_action=act;}
   //--- standart çizelge olayları işleyicisi
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
     {
      if(m_action!=NULL && lparam==Id())
        {
         //--- özel işleyiciyi çağır
         m_action(sparam,(int)lparam);
         return(true);
        }
      else
      //--- CButton ana sınıfından çağrılan işleyicinin sonucunu göster
         return(CButton::OnEvent(id,lparam,dparam,sparam));
     }
  };
//+------------------------------------------------------------------+
//| CControlsDialog sınıfı                                           |
//| Amaç: uygulamanın yönetilmesi için kullanılacak grafik panel  |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CArrayObj         m_buttons;                    // düğme dizisi
public:
                     CControlsDialog(void){};
                    ~CControlsDialog(void){};
   //--- oluştur
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- düğmeyi ekle
   bool              AddButton(MyButton &button){return(m_buttons.Add(GetPointer(butto
protected:
   //--- düğmeleri oluştur
   bool              CreateButtons(void);
  };
//+------------------------------------------------------------------+
//| CControlsDialog nesnesini çizelgede oluştur                      |
//+------------------------------------------------------------------+
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());
//---
  }
//+------------------------------------------------------------------+
//| Düğmeleri oluşturup ControlsDialog paneline ekle                 |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButtons(void)
  {
```

```
//--- düğme koordinatlarını hesapla
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2;
   int y2=y1+BUTTON_HEIGHT;
//--- düğme nesnelerini fonksiyon işaretçilerine ekle
   AddButton(new MyButton("Open",Open));
   AddButton(new MyButton("Save",Save));
   AddButton(new MyButton("Close",Close));
//--- düğmeleri grafiksel olarak oluştur
   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("%s %d düğmesi oluşturulamadı",b.Text(),i);
         return(false);
        }
      //--- her düğmeyi CControlsDialog taşıyıcısına ekle
      if(!Add(b))
         return(false);
     }
//--- başarılı
   return(true);
  }
//--- program başlatıldığında otomatik olarak oluşturulması için nesneyi global seviye
CControlsDialog MyDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- şimdi nesneyi çizelgede oluştur
   if(!MyDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- uygulamayı çalıştır
   MyDialog.Run();
//--- uygulama başarıyla çalıştırıldı
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- iletişim kutusunu yok et
   MyDialog.Destroy(reason);
```

```
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // olay tanıtıcısı
                  const long& lparam,   // long tipli olay parametresi
                  const double& dparam, // double tipli olay parametresi
                  const string& sparam) // string tipli olay parametresi
  {
//--- çizelge olayları için gereken işleyiciyi (CAppDialog) ana sınıftan çağır
   MyDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Ayrıca bakınız

Değişkenler, Fonksiyonlar

Nesne İşaretçileri

MQL5'te karmaşık türde nesnelerin dinamik olarak oluşturulması mümkündür. Bu, oluşturulan nesnenin açıklayıcısını geri döndüren "new" operatörü kullanılarak yapılır. Açıklayıcı boyutu 8 bayttır. Sözdizimsel olarak, MQL5'teki nesne açıklayıcıları, C++'taki işaretçilere benzerdir.

Örnek:

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

C++'tan farklı olarak, yukarıdaki örnekteki "hobject" değişkeni belleğe olan bir işaretçi değildir, bir nesne açıklayıcısıdır. Ayrıca, MQL5'te fonksiyon parametrelerindeki tüm nesnelerin geçişi referans yoluyla yapılmalıdır. Aşağıda, fonksiyon parametrelerindeki nesnelerin referans yoluyla geçişlerine örnekler gösterilmektedir:

```
class Foo
  {
public:
   string            m_name;
   int               m_id;
   static int        s_counter;
   //--- yapıcılar ve yıkıcılar
                     Foo(void){Setup("noname");};
                     Foo(string name){Setup(name);};
                    ~Foo(void){};
   //--- Foo nesnesini başlat
   void              Setup(string name)
     {
      m_name=name;
      s_counter++;
      m_id=s_counter;
     }
  };
int Foo::s_counter=0;
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- otomatik oluşturma ile nesneyi bir değişken olarak bildir
   Foo foo1;
//--- nesnenin geçişini referans yoluyla yap
   PrintObject(foo1);

//--- nesneye bir işaretçi bildir ve onu "new" operatörünü kullanarak oluştur
   Foo *foo2=new Foo("foo2");
//--- nesneye olan işaretçinin geçişini referans yoluyla yap
   PrintObject(foo2); // nesneye olan işaretçi, derleyici tarafından otomatik olarak

//--- Foo nesneleri dizisi bildir
```

```
   Foo foo_objects[5];
//--- nesne dizisinin geçişini yap
   PrintObjectsArray(foo_objects); // nesne dizisinin geçişini yapmak için ayrı bir fo

//--- Foo türü nesnelere olan işaretçi dizisi bildir
   Foo *foo_pointers[5];
   for(int i=0;i<5;i++)
      foo_pointers[i]=new Foo("foo_pointer");
//--- işaretçi dizisinin geçişini yap
   PrintPointersArray(foo_pointers); // işaretçi dizisinin geçişini yapmak için ayrı k

//--- sonlandırmadan önce, işaretçi olarak oluşturulan nesneleri sildiğinizden emin ol
   delete(foo2);
//--- işaretçi dizisini sil
   int size=ArraySize(foo_pointers);
   for(int i=0;i<5;i++)
      delete(foo_pointers[i]);
//---
  }
//+------------------------------------------------------------------+
//|  Nesnelerin geçişi her zaman referansla yapılır                  |
//+------------------------------------------------------------------+
void PrintObject(Foo &object)
  {
   Print(__FUNCTION__,": ",object.m_id," nesne adı=",object.m_name);
  }
//+------------------------------------------------------------------+
//| Nesne dizisinin geçişini yap                                     |
//+------------------------------------------------------------------+
void PrintObjectsArray(Foo &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
      PrintObject(objects[i]);
  }
//+------------------------------------------------------------------+
//| İşaretçi dizisinin geçişini yap                                  |
//+------------------------------------------------------------------+
void PrintPointersArray(Foo* &objects[])
  {
   int size=ArraySize(objects);
   for(int i=0;i<size;i++)
      PrintObject(objects[i]);
  }
//+------------------------------------------------------------------+
```

Kullanmadan önce işaretçiyi kontrol edin

Geçersiz bir işaretçiye erişme girişimi, programın kritik olarak sonlandırılmasına neden olur. CheckPointer fonksiyonu, kullanmadan önce işaretçiyi kontrol etmek için kullanılır. İşaretçi aşağıdaki durumlarda geçersiz olabilir:

- işaretçi NULL'a eşittir;
- nesne, delete operatörü kullanılarak silinmiştir.

Sıfırdan farklı bir değer, verilere ilgili işaretçiden erişilebileceğini ifade eder.

```
class CMyObject
 {
protected:
  double             m_value;
public:
                     CMyObject(void);
                     CMyObject(double value) {m_value=value;};
                    ~CMyObject(void){};
  //---
  double             Value(void) {return(m_value);}
 };
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- başlatılmamış bir nesne oluştur
  CMyObject *pointer;
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("1. işaretçi: ",EnumToString(CheckPointer(pointer)));
  else
    Print("1. pointer.Value()=",pointer.Value());

//--- işaretçiyi başlat
  pointer=new CMyObject(M_PI);
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("2. işaretçi: ",EnumToString(CheckPointer(pointer)));
  else
    Print("2. pointer.Value()=",pointer.Value());

//--- nesneyi sil
  delete(pointer);
  if(CheckPointer(pointer)==POINTER_INVALID)
    Print("3. işaretçi: ",EnumToString(CheckPointer(pointer)));
  else
    Print("3. pointer.Value()=",pointer.Value());
 }
/*
   1. işaretçi: POINTER_INVALID
   2. pointer.Value()=3.141592653589793
   3. işaretçi: POINTER_INVALID
```

```
*/
```

İşaretçiyi hızlı bir şekilde doğrulamak adına, CheckPointer fonksiyonunun örtük çağrısı aracılığıyla işaretçinin geçerliliğini kontrol eden "!" (LNOT) operatörünü de kullanabilirsiniz. Bu, daha kısa ve net bir kod yazımı sağlar. Önceki örnekteki kontroller "!" operatörüyle bu kez şu şekilde görünür:

```
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- başlatılmamış bir nesne oluştur
  CMyObject *pointer;
  if(!pointer)
    Print("1. işaretçi: ",EnumToString(CheckPointer(pointer)));
  else
    Print("1. pointer.Value()=",pointer.Value());

//--- işaretçiyi başlat
  pointer=new CMyObject(M_PI);
  if(!pointer)
    Print("2. işaretçi: ",EnumToString(CheckPointer(pointer)));
  else
    Print("2. pointer.Value()=",pointer.Value());

//--- nesneyi sil
  delete(pointer);
  if(!pointer)
    Print("3. işaretçi: ",EnumToString(CheckPointer(pointer)));
  else
    Print("3. pointer.Value()=",pointer.Value());
 }
/*
   1. işaretçi: POINTER_INVALID
   2. pointer.Value()=3.141592653589793
   3. işaretçi: POINTER_INVALID
*/
```

"==" operatörü, NULL varlığının hızlı kontrolü için kullanılır. Örneğin: ptr==NULL veya ptr!=NULL.**Ayrıca Bakınız**

Değişkenler, Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Yaratılması ve Silinmesi

Referanslar: & Şekillendiricisi ve Anahtar Sözcük this

Parametrelerin Referansla Geçirilmesi

MQL5 içinde basit tipli parametreler değer veya referans ile geçirilirken, bileşik tipli parametreler her zaman referans ile geçirilir. Derleyiciye parametrenin referans ile geçirildiği bilgisini vermek için, parametre isminin önüne ampresan karakteri & eklenir.

Parametrenin referans ile geçirilmesi, değişkenin adresinin geçirilmesi anlamına gelir. Bu yüzden referansla geçirilen parametrelerde yapılan değişiklikler, anında kaynak değişkene yansıyacaktır. Referans ile geçen bir parametre kullanarak, Fonksiyonların çeşitli sonuçlarına aynı anda dönüş yaptırabilirsiniz. Referansla geçirilen parametrelerin değişimini önlemek için const şekillendiricisini kullanabilirsiniz.

Bu şekilde, eğer bir fonksiyonun giriş parametresi bir dizi, bir yapı veya sınıf nesnesi ise, '&' sembolü, fonksiyon isminde değişken tipinden sonra ve kendi isminden önce yer alacaktır.

Örnek

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

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

Yukarıdaki örnekte, double tipli private m_array[] üye - dizisini içeren CDemoClass sınıfı bildirilmiştir. array[] dizisinin referansla geçirildiği setArray()fonksiyonu da bildirilmiştir. Eğer fonksiyon başlığında referansla geçirme yapılmamışsa, yani ampresan karakteri yoksa, kodun derlenmesi sırasında bir hata mesajı oluşturulur.

Dizinin referans ile geçirilmiş olduğu gerçeğine rağmen, bir diziyi diğerine atayamayız. Kaynak dizinin içeriğini hedef diziye eleman bazında kopyalamamız gerekir. Ampresan & karakterinin fonksiyon tanımındaki varlığı, fonksiyon parametresi şeklinde geçirilen dizi ve yapılar için zorunlu bir koşuldur.

Anahtar Sözcük this

Sınıf tipli bir değişken (nesne) hem referans ile hem de işaretçi ile geçirilebilir. Tıpkı referans gibi işaretçi de bir nesneye erişim sağlamaya yarar. Nesne işaretçisinin bildirilmesinden sonra, bunun oluşturulması ve başlatılması için new operatörü kullanılmalıdır.

this rezerve sözcüğü nesnenin, sınıf veya yapı yöntemleri içinde mevcut olan referansını kendine alması için düşünülmüştür. **this** sözcüğü, kullanıldığı yöntemde her zaman nesneye referans yapar ve GetPointer(this) ifadesi, üyesi GetPointer() çağrısını gerçekleştiren fonksiyon olan nesnenin işaretçisini verir. MQL5 dilinde fonksiyonlar nesnelere dönüş yapamazlar ama nesne işaretçilerine dönüş yapabilirler.

Bu şekilde, nesneye dönüş yapacak bir fonksiyona ihtiyacımız varsa, nesne işaretçisine GetPointer(this) biçiminde dönüş yapabiliriz. CDemoClass tarifinin içine sınıfın nesnesinin işaretçisine dönüş yapan getDemoClass() fonksiyonunu ekleyelim.

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

public:
   void              setArray(double &array[]);
   CDemoClass       *getDemoClass();
  };
//+------------------------------------------------------------------+
//| dizinin doldurulması                                             |
//+------------------------------------------------------------------+
void  CDemoClass::setArray(double &array[])
  {
   if(ArraySize(array)>0)
     {
      ArrayResize(m_array,ArraySize(array));
      ArrayCopy(m_array,array);
     }
  }
//+------------------------------------------------------------------+
//| kendi işaretçisine dönüş yapar                                   |
//+------------------------------------------------------------------+
CDemoClass *CDemoClass::getDemoClass(void)
  {
   return(GetPointer(this));
  }
```

Yapıların işaretçileri yoktur. *new*, *delete* operatörleri ve GetPointer(this) yapılara uygulanamaz.

Ayrıca Bakınız

Nesne İşaretçileri, Nesnelerin Yaratılması ve Silinmesi, Değişkenlerin Görünürlük Alanları ve Ömürleri

İşlemler ve İfadeler

Bazı karakterlerin ve karakter dizgilerinin özel bir önemi vardır. Bunlara işlem sembolleri denir, örneğin:

```
+ - * / %     Aritmetik işlemlerin sembolleri
&& ||         Mantıksal işlemler için semboller
= += *=       Karakter atama operatörleri
```

İşlem sembolleri ifadelerde kullanılır ve işlenen terimler uygun olarak verildiği zaman anlam kazanırlar. Noktalama işaretleri gibi vurgulanırlar. Bunlar parantezler, çengel parantezler, virgül, iki nokta ve noktalı virgüldür.

İşlem sembolleri, noktalama işaretleri ve boşluklar, dil elemanlarını birbirinden ayırmak için kullanılır.

Bu bölüm, şu konu başlıkları için açıklamalar içerir:

- İfadeler
- Aritmetik İşlemler
- Atama İşlemleri
- İlinti İşlemleri
- Mantıksal İşlemler
- Bitsel İşlemler
- Diğer İşlemler
- Özellikler ve İşlem Sıraları

İfadeler

İfadeler bir veya daha fazla işlenenden ve işlem sembolünden oluşur. Birkaç satırda yazılabilirler.

Örnekler:

```
a++; b = 10;          // bir satıra yerleştirilmiş birkaç ifade
//---  birkaç satıra bölünmüş bir ifade
x = (y * z) /
    (w + 2) + 127;
```

Noktalı virgül ile biten ifadeler (;) bir operatördür.

Ayrıca Bakınız

Öncelik Kuralları

Aritmetik İşlemler

Aritmetik işlemler toplamsal ve çarpımsal işlemleri içerirler:

```
Değişkenlerin toplamı                  i = j + 2;
Değişkenlerin farkı                    i = j - 3;
İşaret değiştirme                      x = - x;
Değişkenlerin çarpımı                  z = 3 * x;
Bölme                                  i = j / 5;
Bölümden kalan                         dakikalar = zaman % 60;
Değişken değerine 1 ekleme             i++;
Değişken değerine 1 ekleme             ++i;
Değişken değerinden 1 çıkarma          k--;
Değişken değerinden 1 çıkarma          --k;
```

Artırma ve eksiltme işlemleri sadece değişkenlere uygulanabilir, sabitlerde kullanılamaz. Ön ekli artırma (++i) ve eksiltme (--k), değişkenin kullanımından hemen sonra değişkene uygulanır .

Son ekli artırma (i++) ve eksiltme (k--), ifade içinde değişkenin kullanımından hemen sonra değişkene uygulanır.

Önemli Uyarı

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

Yukarıdaki ifadenin bir programlama ortamından diğerine taşınması sonucunda, hesaplamaya dair sorunlar çıkabilir (örneğin Borland C++'dan MQL5'e). Genel olarak hesaplamaların sırası derleyici uygulamasına bağlıdır. Pratikte, son ekli artırmayı (eksiltmeyi) uygulamanın iki yolu vardır:

1. Son ekli artırma (eksiltme) tüm ifade hesaplandıktan sonra değişkene uygulanır.
2. Son ekli artırma (eksiltme) işlem anında hemen değişkene uygulanır.

Mevcut durumda MQL5 içinde son ekli artırma (eksiltme) yolu uygulanmaktadır. Ama bu özelliğin bilinmesi durumunda bile kullanımının denenmesi tavsiye edilmez

Örnekler:

```
int a=3;
a++;            // geçerli ifade
int b=(a++)*3;  // geçersiz ifade
```

Ayrıca Bakınız

Öncelik Kuralları

Atama İşlemleri

Verilen işlemi içeren ifadenin değeri, atamadan sonra soldaki işlenenin değeri olur:

```
x değerini, y değişkenine atama                                    y = x;
```

Aşağıdaki işlemler aritmetik veya bitsel işlemleri, atama işlemiyle birleştirir:

```
y değişkenine x değerini ekleme                                    y += x;
y değişkeninden x değerini çıkarma                                 y -= x;
y değişkenini x ile çarpma                                         y *= x;
y değerini x'e bölme                                               y /= x;
y değişkeninin, x'e bölümünden kalan                               y %= x;
y ikili ifadesinin x bit sağa kaydırılması                         y >>= x;
y ikili ifadesinin x bit sola kaydırılması                         y <<= x;
y ve x ikili gösterimi için VE bitsel işlemi                       y &= x;
y ve x ikili gösterimi için VEYA bitsel işlemi                     y |= x;
y ve x ikili gösteriminde VEYA işleminin özelleşmesi (XOR işlemi)  y ^= x;
```

Bitsel işlemler sadece tamsayılara uygulanabilir. y gösteriminin mantıksal olarak x bit sağa/sola kaydırılması işlemi için, x değerinin en küçük 5 basamağı kullanılır ve en yüksek basamaklar bırakılır. Yani kaydırma 0-31 arasındaki bitlere yapılır.

%= işleminde (x modülünde y değeri) sonuç işareti bölünen sayının işareti ile aynıdır.

Atama işlemi bir ifade içinde birkaç defa kullanılabilir. Bu durumda ifadenin işlenmesi soldan sağa doğru gerçekleşecektir:

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

İlk olarak 3 değeri x değişkenine atanır, ardından x değeri (yani 3) y değişkenine atanır.

Ayrıca Bakınız

Öncelik Kuralları

İlinti İşlemleri

Mantıksal 'YANLIŞ' sıfır değeriyle temsil edilirken, mantıksal 'DOĞRU' sıfır olmayan herhangi bir değer ile temsil edilir.

İlinti işlemleri veya mantıksal işlemler içeren ifadeler 'YANLIŞ' (0) veya 'DOĞRU'dur (1).

```
a, b'ye eşitse doğru                          a == b;
a, b'ye eşit değilse doğru                    a != b;
a, b'den küçük ise doğru                      a < b;
a, b'den büyük ise doğru                      a > b;
a, b'den küçük veya b'ye eşit ise doğru       a <= b;
a, b'den büyük veya b'ye eşit ise doğru       a >= b;
```

İki reel sayının eşitliği karşılaştırılamaz. Çoğu durumda, virgülden sonra 15-inci basamaktaki değerler nedeniyle aynı görünen sayılar eşit olmayabilir. İki reel sayıyı doğru şekilde karşılaştırabilmek için bu iki sayının normalleştirilmiş farklarını sıfırla karşılaştırın.

Örnek:

```
bool CompareDoubles(double number1,double number2)
  {
   if(NormalizeDouble(number1-number2,8)==0) return(true);
   else return(false);
  }
void OnStart()
  {
   double first=0.3;
   double second=3.0;
   double third=second-2.7;
   if(first!=third)
     {
      if(CompareDoubles(first,third))
         printf("%.16f ve %.16f eşittir",first,third);
     }
  }
// Sonuç: 0.3000000000000000  0.2999999999999998   sayıları eşittir.
```

Ayrıca Bakınız

Öncelik Kuralları

Mantıksal İşlemler

Mantıksal Red 'DEĞİL' (!)

Mantıksal red işleminin (!) işleneni aritmetik tipte olmalıdır. İşlenen değer 'YANLIŞ' (0) ise sonuç 'DOĞRU' (1) olur, İşlenen değer 'YANLIŞ' (0) değilse sonuç 'YANLIŞ' (0) olur .

```
if(!a) Print("'a' değildir");
```

Mantıksal İşlem 'VEYA' (||)

x ve y değerlerinin mantıksal 'VEYA' (||) işlemi. Eğer x veya y değerleri doğruysa (boş değilse), ifade değeri 'DOĞRU' (1) olur. Aksi durumda - 'YANLIŞ' (0).

```
if(x<0 || x>=max_bars) Print("kapsam dışı");
```

Mantıksal İşlem 'VE' (&&)

x ve y değerlerinin mantıksal 'VE' (&&) işlemi. Eğer x ve y değerleri doğruysa (boş değilse), ifade değeri 'DOĞRU' (1) olur. Aksi durumda - 'YANLIŞ' (0).

Mantıksal İşlemlerinin Özet Tahmin

"Özet tahmin" şeklinde isimlendirilen taslak mantıksal işlemlerde uygulanır. İfadenin sonucu tam olarak tahmin edildiğinde ifadenin hesaplama kısmı silinir.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ilk kısa tahmin örneği
   if(func_false() && func_true())
     {
      Print("İşlem &&: Bu ifadeyi hiçbir zaman görmeyeceksin");
     }
   else
     {
      Print("İşlem &&: İlk tahminin sonucu yanlış, bu yüzden ikinci hesaplanmadı");
     }
//--- ikinci kısa tahmin örneği
   if(!func_false() || !func_true())
     {
      Print("İşlem ||: İlk ifadenin sonucu doğru, bu yüzden ikincisi hesaplanmayacak")
     }
   else
     {
      Print("İşlem ||: Bu ifadeyi hiçbir zaman görmeyeceksin");
     }
  }
//+------------------------------------------------------------------+
```

```
//| her zaman false (yanlış) dönüşü yapan fonksiyon                  |
//+------------------------------------------------------------------+
bool func_false()
  {
   Print("Fonksiyon func_false()");
   return(false);
  }
//+------------------------------------------------------------------+
//| her zaman true (doğru) dönüşü yapan fonksiyon                    |
//+------------------------------------------------------------------+
bool func_true()
  {
   Print("Fonksiyon func_true()");
   return(true);
  }
```

Ayrıca Bakınız

Öncelik Kuralları

Bitsel İşlemler

Bire tamamlama

Değişken değerinin bire tamamlanmasıdır. İfade değerinin haneleri, değişken değerinin 0 içeren tüm hanelerinde 1, değişken değerinin 1 içeren tüm hanelerinde ise 0 olacak şekilde ayarlanır.

```
b = ~n;
```

Örnek:

```
   char a='a',b;
   b=~a;
   Print("a = ",a, "  b = ",b);
// Sonuç şöyle olur:
// a = 97   b = -98
```

Sağa Kaydırma

x değerinin sağa doğru y basamak kaydırılmasının ikili gösterimi. Kaydırma değerinin tipi işaretsiz ise mantıksal sağ kaydırma yapılır, yani serbest kalan sol taraf sıfırlarla doldurulur.

Kaydırma değerinin tipi işaretli ise aritmetik sağ kaydırma yapılır, yani serbest kalan sol taraftaki haneler işaret bitinin değeriyle doldurulur (işaret bitinin değeri sayı pozitifse 0, sayı negatifse 1 olur).

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

Örnek:

```
   char a='a',b='b';
   Print("Önce:  a = ",a, "  b = ",b);
//--- sağa kaydırma
   b=a>>1;
   Print("Sonra:   a = ",a, "  b = ",b);
// Sonuç şöyle olur:
// Önce:  a = 97   b = 98
// Sonra:   a = 97   b = 48
```

Sola Kaydırma

İkili x ifadesi y basamak sola kaydırılır. Boşaltılan sağ taraftaki haneler ise sıfırla doldurulur.

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

Örnek:

```
   char a='a',b='b';
   Print("Önce:  a = ",a, "  b = ",b);
//--- sola kaydır
   b=a<<1;
   Print("Sonra:   a = ",a, "  b = ",b);
// Sonuç şöyle olur:
// Önce:  a = 97   b = 98
```

```
// Sonra:   a = 97   b = -62
```

Kaydırma değerindeki bitlerin sayısı kaydırılan değişkenin uzunluğuna eşit veya daha büyük ise kaydırma işlemi tavsiye edilmez, böyle bir işlemin sonucu tanımsızdır.

Bitsel 'VE' İşlemi

İkili olarak kodlanmış x ve y gösterimlerinin bitsel 'VE' işlemi. İfade, x ve y değişkenlerinin her ikisinin de sıfır içermediği tüm hanelerde 1 (DOĞRU) değerini alırken, diğer tüm hanelerde 0 (YANLIŞ) değerini alır.

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

Örnek:

```
   char a='a',b='b';
//--- VE işlemi
   char c=a&b;
   Print("a = ",a,"  b = ",b);
   Print("a & b = ",c);
// Sonuç şöyle olur:
// a = 97   b = 98
// a & b = 96
```

Bitsel 'VEYA' İşlemi

x ve y ikili ifadelerinin bitsel 'VEYA' işlemi. İfade değeri, x ve y değerlerinin 0 içerdiği tüm hanelerde 1 değerini alırken, diğer tüm hanelerde ise 0 değerini alır.

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

Örnek:

```
   char a='a',b='b';
//--- VEYA işlemi
   char c=a|b;
   Print("a = ",a,"  b = ",b);
   Print("a | b = ",c);
// Sonuç şöyle olur:
// a = 97   b = 98
// a | b = 99
```

Bitsel 'Özel VEYA' işlemi

İkili x ve y ifadesinin bitsel 'özel VEYA' (eXclusive OR) işlemi. İfade değeri, x ve y değişkenlerinin farklı değerler aldığı tüm hanelerde 1, diğer tüm hanelerde ise 0 olur.

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

Örnek:

```
   char a='a', b='b';
//--- Özelleştirilen VEYA işlemi
```

```
   char c=a^b;
   Print("a = ",a,"  b = ",b);
   Print("a ^ b = ",c);
// Sonuç şöyle olur:
// a = 97   b = 98
// a ^ b = 3
```

Bitsel işlemler sadece tamsayılarla gerçekleştirilir.

Ayrıca Bakınız

Öncelik Kuralları

Diğer işlemler

İndisleme ([])

Dizinin i-inci elemanı adreslenirken ifade değeri 'i' numaralı değişkenin değeridir.

Örnek:

```
array[i] = 3; // 3 değerini i-inci dizi elemanına ata.
```

Dizi indisleri sadece tamsayı olabilir. Dizilerde dört boyut ve aşağısına izin verilir. Her boyut, 0 'dan "**boyut büyüklüğü**-1" değerine kadar indislenir. Örneğin, 50 elemanlı tek boyutlu bir dizi için ilk elemana yapılan referans 'dizi[0]' şeklinde, son elemana yapılan referans ise 'dizi[49]' şeklinde yazılır.

Dizi büyüklüğünün ötesine adresleme yapıldığında, yürütme alt sistemi bir hata oluşturur ve program durur.

x1, x2 ,..., xn Argümanlarına Sahip Fonksiyonun Çağrılması

Her argüman bir sabiti, bir değişkeni veya bir ifadeyi temsil edebilir. Geçirilen argümanlar virgüllerle ayrılmalı ve parantezler içinde olmalıdır, açılan parantez çağrılan fonksiyonun isminden sonra gelmelidir.

İfade değeri fonksiyon tarafından dönüş yapılan değerdir. Eğer fonksiyonun dönüş tipi 'void' ise, bu fonksiyon atama işleminde sağ tarafa yerleştirilemez. x1,..., xn ifadelerinin tam olarak bu sırayla çalıştırıldığını lütfen not edin.

Örnek:

```
   int length=1000000;
   string a="a",b="b",c;
//---Diğer işlemler
   int start=GetTickCount(),stop;
   long i;
   for(i=0;i<length;i++)
     {
      c=a+b;
     }
   stop=GetTickCount();
   Print("'c = a + b' için zaman = ",(stop-start)," milisaniye, i = ",i);
```

Virgül İşlemi (,)

Virgül ile ayrılmış ifadeler soldan sağa doğru çalıştırılırlar. Soldaki ifadenin hesabının tüm etkileri sağdaki ifade çalıştırılmadan görülebilir. Sonuç tipi ve değeri sağdaki ifadenin tip ve değeriyle örtüşür. Geçirilecek parametrelerin listesi (yukarıya bakın) örnek olarak gösterilebilir.

Örnek:

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

Nokta İşlemi (.)

Yapı ve sınıfların genele açık üyelerine doğrudan erişim (public üyelere erişim) sağlamak için nokta işlemi kullanılır. Sözdizim:

```
Yapı_tipi_degiskeninin_ismi.Üye_ismi
```

Örnek:

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

Kapsam Çözünürlüğü İşlemi (::)

Her fonksiyon mql5 içinde kendi çalışma kapsamına sahiptir. Örneğin, Print() sistem fonksiyonu global kapsamda çalıştırılır. İçe aktarılmış fonksiyonlar, ilgili içe aktarım kapsamında çağrılır. Sınıf yöntemleri ilgili sınıfın kapsamına sahiptir. Kapsam çözünürlük işlemi için sözdizim şu şekildedir:

```
[Kapsam_ismi]::Fonksiyon_ismi(parametreler)
```

Eğer kapsam ismi yoksa fonksiyon global kapsamda aranır. Eğer hiç kapsam çözünürlük işlemi yoksa fonksiyon en yakın kapsamda aranır. Eğer yerel kapsamda hiç fonksiyon yoksa arama global kapsama yöneltilir.

Kapsam çözünürlük işlemi sınıf fonksiyonunu tanımlamak için de kullanılır.

```
tip Sınıf_ismi::Fonksiyon_ismi(parametre_açıklaması)
   {
// fonksiyon gövdesi
   }
```

Aynı isme ve farklı uygulama içeriklerine sahip olan fonksiyonların aynı program üzerinde çalıştırılması belirsizlik oluşturur. Açık kapsam özellikleri olmadan yapılan fonksiyon çağrılarının sıralama önceliği şu şekildedir:

1. Sınıf yöntemleri. Belirtilen isimde bir fonksiyon sınıf bünyesinde yoksa sonraki aşamaya geç.
2. MQL5 foksiyonları. Kullanılan dil böyle bir fonksiyon içermiyorsa sonraki aşamaya geç.
3. Kullanıcı tanımlı global fonksiyonlar. Belirtilen isimde bir fonksiyon bulunmuyorsa sonraki aşamaya geç.

4. İçe-aktarılan fonksiyonlar. Belirtilen isimde bir fonksiyon bulunmuyorsa, derleyici hata dönüşü yapacaktır.

Fonksiyon çağrılarında belirsizliği önlemek için, kapsam çözünürlüğü işlemini kullanarak fonksiyonun kapsamını açık bir şekilde belirtin.

Örnek:

```
#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("ebeveyn GetLastError",CCheckContext::GetLastError());
   ::Print("bizim GetLastError",GetLastError());
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   CCheckContext2 test;
   test.Print();
  }
//+------------------------------------------------------------------+
```

Veri Tipinin Büyüklüğün Alınması veya Herhangi bir Veri Tipi Nesnesinin Büyüklüğü (sizeof)

sizeof işlemi ile bir tanımlayıcıya veya bir veri tipine karşılık gelen bellek büyüklüğü alınabilir. sizeof işlemi aşağıda gösterilmiştir:

Örnek:

```
sizeof(ifade)
```

Her tanımlayıcı veya parantez içine iliştirilmiş her tip ismi buradaki ifadenin yerine kullanılabilir. Void tip isminin kullanılamayacağı, tanımlayıcının bit alanına ait olamayacağı veya tanımlayıcının bir fonksiyon ismi olamayacağı bilgileri not edilmelidir.

Eğer ifade bir statik dizinin ismi ise (yani birinci boyut veri ise) sonuç tüm dizinin büyüklüğü olacaktır (yani elemanların sayısı ile tip uzunluğunun çarpımı). Eğer ifade bir dinamik dizinin ismi ise (ve ilk boyut belirtilmemişse) sonuç dinamik dizi nesnesinin büyüklüğü olacaktır.

İşlem bir yapı veya sınıf tipine veya bunların tanıtıcılarına uygulandığında, sonuç bu yapı veya sınıfın gerçek büyüklüğü olacaktır.

Örnek:

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

Büyüklük değeri derleme aşamasında hesaplanır.

Ayrıca bakınız

Öncelik Kuralları

Öncelik Kuralları

Tabloda yer alan tüm işlem grupları aynı önceliğe sahiptir. Grubunun tablodaki konumu ne kadar yüksekse, işlemin önceliği de o kadar yüksektir. Öncelik kuralları, işlemlerin ve işlenenlerin gruplandırılma şeklini belirler.

Uyarı: MQL5 dilindeki işlem önceliği, C++ dilinde kabul edilmiş önceliğe karşılık gelir ve MQL4 dilindeki önceliklerden farklıdır.

İşlem	Açıklama	Çalıştırılma Sırası
() [] .	Fonksiyon Çağrısı Bir dizi elemanına referans verme Bir yapı elemanına referans verme	Soldan sağa
! ~ - ++ -- (tip) sizeof	Mantıksal red Bitsel red (tamamlayıcı) İşaret değiştirme Bir artırma Bir eksiltme Tip Dönüşümü Büyüklüğü bitlerle belirleme	Sağdan sola
* / %	Çarpma Bölme Modül bölümü	Soldan sağa
+ -	Toplama Çıkarma	Soldan sağa
<< >>	Sola kaydırma Sağa kaydırma	Soldan sağa
< <= > >=	-den küçük -den küçük veya eşit -den büyük -den büyük veya eşit	Soldan sağa
== !=	Eşittir Eşit değildir	Soldan sağa
&	Bitsel VE işlemi	Soldan sağa
^	Bitsel özel VEYA	Soldan sağa
|	Bitsel VEYA işlemi	Soldan sağa
&&	Mantıksal VE işlemi	Soldan sağa
||	Mantıksal VEYA işlemi	Soldan sağa
?:	Koşullu Operatör	Sağdan sola
= *= /=	Atama Atamayla çarpma Atamayla bölme	Sağdan sola

İşlem	Açıklama	Çalıştırılma Sırası
%= += -= <<= >>= &= ^= |=	Atamayla modül Atamayla toplama Atamayla çıkarma Atamayla sola kaydırma Atamayla sağa kaydırma Atamayla bitsel VE Atamayla bitsel özel VEYA Atamayla bitsel VEYA	
,	Virgül	Soldan sağa

İşlemin çalıştırılma sırasını değiştirmek için, daha yüksek öncelikten parantezler kullanılır.

Operatörler

Dil operatörleri, bir görevi başarmak için çalıştırılması gereken bazı algoritmik işlemleri tarif eder. Program gövdesi, bu gibi bir dizi operatörden oluşur. Birbirini takip eden operatörler, noktalı virgül ile ayrılır.

Operatör	Açıklama
Birleşik operatör {}	Çengel parantez içine iliştirilmiş, bir veya daha fazla operatör {}
İfade operatörü (;)	Noktalı virgül (;) ile biten her ifade
return operatörü	Mevcut fonksiyonu sonlandırır ve kontrolü çağrılan programa döndürür
if-else koşullu operatör	Seçim yapmak gerektiğinde kullanılır
?: koşullu operatör	if-else koşullu operatörünün basit bir benzeri
switch seçim operatörü	Kontrolü ifade değerine karşılık gelen operatöre geçirir
while döngü operatörü	Bir operatörü, kontrol edilen ifade yanlış olana kadar gerçekleştirir. İfade her tekrar öncesi kontrol edilir
for döngü operatörü	Bir operatörü, kontrol edilen ifade yanlış olana kadar gerçekleştirir. İfade her tekrar öncesi kontrol edilir
do-while döngü operatörü	Bir operatörü, kontrol edilen ifade yanlış olana kadar gerçekleştirir. Her tekrardan sonra son koşul kontrol edilir. Döngü gövdesi en az bir kere çalıştırılır.
break operatörü	İliştirildiği en yakın dışsal döngü operatörünü (switch, while, do-while veya for) sonlandırır
continue operatörü	Kontrolü, en yakın dışsal döngü operatörünün (switch, while, do-while veya for) başlangıcına geçirir
new operatörü	Uygun büyüklükte bir nesne oluşturur ve oluşturulan nesnenin bir tarifcisine dönüş yapar.
delete operatörü	new operatörü ile oluşturulmuş nesneyi siler

Bir operatör, bir veya daha fazla satırı işgal edebilir. İki veya daha fazla operatör aynı satırda yer alabilir. İfade emrini kontrol eden operatörler (if, if-else, switch, while ve for), birbirleriyle iç içe olabilirler.

Örnek:

```
if(Month() == 12)
  if(Day() == 31) Print("Mutlu yıllar!");
```

Ayrıca Bakınız

Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Yaratılması ve Silinmesi

Birleşik Operatör

Bir birleşik operatör (bir blok) parantezler {} içine iliştirilmiş herhangi tipte bir veya daha fazla operatörden oluşur. Kapanış parantezinin ardından noktalı virgül (;) gelmemelidir.

Örnek:

```
if(x==0)
  {
   Print("geçersiz pozisyon x = ",x);
   return;
  }
```

Ayrıca Bakınız

Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Yaratılması ve Silinmesi

İfade Operatörü

Noktalı virgül ile (;) tamamlanan tüm ifadeler bir operatördür. Burada ifade operatörlerine bazı örnekler verilmiştir.

Atama Operatörü

Tanımlayıcı = ifade;

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

Atama operatörü ifade içinde defalarca kullanılabilir. Bu durumda ifade sağdan sola doğru işlenir.

Fonksiyon çağırma operatörü

Fonksiyon_ismi (arguman1,..., argumanN);

```
FileClose(dosya);
```

Boş operatör

Sadece bir noktalı virgülden oluşur (;) ve kontrol operatörünün boş gövdesini ifade eder.

Ayrıca bakınız

Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Yaratılması ve Silinmesi

Return Operatörü

'Return' operatörü mevcut fonksiyonun çalışmasını sonlandırır ve kontrolü fonksiyonu çağıran programa verir. Çağrılan fonksiyon hesaplanan ifadenin sonucuna dönüş yapmasını sağlar. İfade (return fonksiyonunun parametresi) bir atama operatörü içerebilir.

Örnek:

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

Dönüş tipi void olan fonksiyonlarda, ifade içermeyen return operatörü kullanılmalıdır:

```
void SomeFunction()
  {
   Print("Merhaba!");
   return;     // bu operatör kaldırılabilir
  }
```

Fonksiyonun sağ küme parantezi, return operatörünün ifadesiz olarak gizlice çalıştırılması anlamına gelir.

Dönüş yapılabilecekler: basit tipler, basit yapılar, nesne işaretçileri. Dizilere, sınıf nesnelerine ve birleşik yapı tipli değişkenlere *return* operatörü ile dönüş yapamazsınız.

Ayrıca Bakınız

Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Yaratılması ve Silinmesi

If-Else Koşullu Operatörü

IF - ELSE (eğer-değilse) operatörü bir seçim yapılması gerektiğinde kullanılır. Biçimsel olarak, sözdizimi şu şekildedir:

```
if (ifade)
    operatör1
else
    operatör2
```

Eğer ifade doğruysa operatör1 çalıştırılır ve kontrol, operatör2 sonrasında gelen operatöre verilir (operatör2 çalıştırılmaz). Eğer ifade hatalıysa operatör2 çalıştırılır.

if operatörünün else kısmı atlanabilir. Bu şekilde else kısmı atlanmış olan if operatörlerinde bir farklılaşma görülebilir. Bu durumda gizli else operatörü, aynı bloktaki else kısmı olmayan, en yakın if operatörüne adresleme yapar.

Örnekler:

```
//--- Else (değilse) kısmı ikinci if (eğer) operatörünü gösterir:
if(x>1)
   if(y==2) z=5;
else     z=6;
//--- else kısmı ilk if operatörünü gösterir:
if(x>1)
  {
   if(y==2) z=5;
  }
else        z=6;
//--- İç içe operatörler
if(x=='a')
  {
   y=1;
  }
else if(x=='b')
  {
   y=2;
   z=3;
  }
else if(x=='c')
  {
   y=4;
  }
else Print("HATA");
```

Ayrıca Bakınız

Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Yaratılması ve Silinmesi

Üçlü Operatör ?:

Üçlü operatörün genel şekli şu şekildedir:

```
ifade1 ? ifade2 : ifade3
```

İlk işlenen - "ifade1" - için bir bool tipi ile sonuçlanacak her ifade kullanılabilir. Sonuç true (doğru) ise, operatör ikinci işlenen tarafından ayarlanır, yani "ifade2" çalıştırılır.

İlk işlenen false (yanlış) ise, üçüncü işlenen - "ifade3" çalıştırılır. İkinci ve üçüncü işlenenler (yani "ifade2" ve "ifade3") bir tip değerine dönüş yapmalıdır ve bu tip void tipi olmamalıdır. Koşullu operatörün çalıştırılmasının sonucu, ifade1'in sonucuna bağlı olarak ifade2 veya ifade3 olabilir.

```
//--- bir günlük aralık için, açılış ve kapanış fiyatları arasındaki farkı normalize e
double true_range = (High==Low)?0:(Close-Open)/(High-Low);
```

Bu giriş, şuna eşdeğerdir:

```
   double true_range;
   if(High==Low)true_range=0;                // High ve Low eşitse
   else true_range=(Close-Open)/(High-Low); // gerçek aralık boş değilse
```

Operatör Kullanım Kısıtlamaları

"ifade1" değerine bağlı olarak, işlemci iki değerden birine dönüş yapmalıdır (ya "ifade2" ya da "ifade3"). Bu ifadeler için birkaç sınırlama vardır:

1. Kullanıcı tanımlı tipi basit tip ile veya sayım ile karıştırmayın. NULL, işaretçi için kullanılabilir.
2. Değerlerin tipi basitse, operatör en büyük tipe sahip olacaktır (bakınız Tip dönüşümü).
3. Eğer değerlerden biri 'sayım' ise ve diğerinin de sayısal bir tipi varsa; sayım, int tipi ile değiştirilir ve ikinci kural uygulanır.
4. İki değer de sayım ise, tipleri aynı olmalıdır. Operatör sayım tipinde olacaktır.

Kullanıcı tanımlı tipler (sınıflar veya yapılar) için kısıtlamalar:

a) Tipler aynı olmalıdır veya bir diğerinden türetilmiş olmalıdır.

b) Eğer tipler aynı (akraba) değilse, çocuk tip gizlice ebeveyne dönüştürülür. Yani operatör ebeveynin tipinde olur.

c) Nesne ve işaretçiyi karıştırmayın - iki ifade de ya nesnedir ya da işaretçidir. NULL, işaretçi için kullanılabilir.

Not

Koşullu operatörü aşırı yüklenmiş fonksiyon için argüman olarak kullanırken dikkatli olun, çünkü bir koşullu operatörün sonucu, program derlendiği zaman tanımlanır. Sonuç tipi, "ifade1" ve "ifade2" tipleri arasından büyük olan şeklinde tanımlanır.

Örnek:

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

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

void OnStart()
  {
   func(Expression2);
   func(Expression3);

   func(Expression1?Expression2:Expression3);   // açık dizgi dönüşümü olduğunda uyar
   func(!Expression1?Expression2:Expression3);  // açık dizgi dönüşümü olduğunda uyar
  }

//   Sonuç:
//   double argüman: 3.141592653589793
//   string argüman: 3.1415926
//   string argüman: 3.141592653589793
//   string argüman: 3.1415926
```

Ayrıca Bakınız

Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Yaratılması ve Silinmesi

Switch Operatörü

İfade değerini tüm '*case*' varyantlarındaki sabitlerle karşılaştırır ve kontrolü ifade değerine karşılık gelen operatöre geçirir. Her bir *case* varyantı, bir tamsayı sabit, bir sözlü sabit veya bir ifade ile işaretlenebilir. Sabit ifade, değişkenleri ve fonksiyon isimlerini içeremez. *Switch* operatörünün ifadesi tamsayı tipinde olmalıdır - int veya uint.

```
switch(ifade)
  {
   case sabit: operatörler
   case sabit: operatörler
     ...
   default: operatörler
  }
```

Eğer *case* operatörleri içindeki sabitlerden hiçbiri ifade değerine eşit değilse, *default* etiketi ile imlenen operatörler çalıştırılır. *default* varyantının bildirilmesi veya sonda yer alması gerekli değildir. İfade değerine ve *default* varyantına karşılık gelen sabitlerin hiçbiri mevcut değilse, hiçbir eylem gerçekleştirilmez.

Sabit içeren '*case*' anahtar sözcüğü sadece bir etikettir ve eğer operatörler bazı '*case*' varyantları için çalıştırılırsa, takip eden tüm varyantların operatörleri *break* operatörü gelene kadar çalıştırılır. Bu, bir dizi operatörün birkaç varyantla bağlanmasını sağlar.

Derleme süresince bir sabit ifade hesaplanır. Bir *switch* operatörü içindeki iki sabit aynı değeri alamaz.

Örnekler:

```
//--- İlk örnek
switch(x)
  {
   case 'A':
      Print("DURUM A");
      break;
   case 'B':
   case 'C':
      Print("CASE B veya C");
      break;
   default:
      Print("A, B veya C DEĞİL");
      break;
  }

//---  İkinci örnek
   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
*/
```

Ayrıca Bakınız

Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Yaratılması ve Silinmesi

Döngü Operatörü while

while operatörü, kontrol edilmiş bir ifadeden ve çalıştırılması gereken bir operatörden meydana gelir:

```
while(ifade)
 operatör;
```

Eğer ifade doğruysa, operatör ifade yanlış olana kadar çalıştırılır. İfade yanlışsa kontrol bir sonraki operatöre geçirilir. İfade değeri operatör çalıştırılmadan belirlenir. Böylece, ifade en başında zaten yanlışsa, operatör hiç çalıştırılmamış olacaktır.

Not

Eğer döngü içinde büyük miktarda tekrar yapılması bekleniyorsa, zorla sonlandırma işlemi, IsStopped() fonksiyonu aracılığı ile kontrol edilebilir.

Örnek:

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

Ayrıca Bakınız

Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Yaratılması ve Silinmesi

For Döngü Operatörü

'For' operatörü, üç ifadeden ve bir çalıştırılabilir operatörden oluşur:

```
for(ifade1; ifade2; ifade3)
  operatör;
```

İfade 1, döngü başlangıcını ifade eder. İfade2, döngü sonlandırma koşullarını kontrol eder. Eğer doğruysa, döngü gövdesi for çalıştırılır. Döngü, ifade2'yi yanlış sonuç alana kadar tekrar eder. Eğer yanlışsa, döngü sonlandırılır ve kontrol bir sonraki operatöre verilir. İfade3, her tekrardan sonra hesaplanır.

Bir for operatörü şu ardışık operatörlere eş değerdir:

```
ifade1;
while(ifade2)
  {
  operatör;
  ifade3;
  };
```

Bu üç ifadelerden herhangi biri veya hiç biri for operatöründe yer almayabilir ama bunları ayıran noktalı virgüller (;) kesinlikle atlanmamalıdır. İfade2 atlandığında kesinlikle doğru olduğu varsayılır. For(;;) operatörü, devamlı bir döngüdür; while(1) operatörüne eşdeğerdir. İfade1 veya ifade3 ifadelerinin her ikisi de virgül ',' ile birleştirilmiş çeşitli ifadeler barındırabilirler.

Not

Eğer döngü içinde büyük miktarda tekrar yapılması bekleniyorsa, zorla sonlandırma işlemi, IsStopped() fonksiyonu aracılığı ile kontrol edilebilir.

Örnekler:

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

Ayrıca Bakınız

Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Yaratılması ve Silinmesi

Döngü Operatörü do while

for ve while döngüleri sonlandırma işlemini başlangıç esnasında kontrol eder, döngünün sonunda değil. Üçüncü döngü operatörü do - while, sonlandırma koşulunu bitişte yani her bir döngü tekrarından sonra kontrol eder. Döngü gövdesi her zaman en az bir defa çalıştırılır.

```
do
  operatör;
while(ifade);
```

İlk olarak operatör çalıştırılır, ardından ifade hesaplanır. Eğer ifade doğruysa, o zaman operatör yeniden çalıştırılır ve böyle devam eder. İfade yanlış değer aldığında döngü sonlandırılır.

Not

Eğer döngü içinde büyük miktarda tekrar yapılması bekleniyorsa, zorla sonlandırma işlemi, IsStopped() fonksiyonu aracılığı ile kontrol edilebilir.

Örnek:

```
//--- Fibonacci serisini hesapla
   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++; // bu operatör olmadan döngü sonsuza kadar devam eder!
     }
   while(i<counterFibonacci && !IsStopped());
```

Ayrıca Bakınız

Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Yaratılması ve Silinmesi

Break Operatörü

Bir break operatörü, en yakın dışsal switch, while, do-while veya for operatörünün çalışmasını sonlandırır. Kontrol, sonlandırılmış olanın ardından gelen operatöre geçirilir. Bu operatörün amaçlarından biri de; belirli bir değer bir değişkene atandığında, döngü işlemini sonlandırmaktır.

Örnek:

```
//--- ilk sıfır değerli elemanın aranması
for(i=0;i<array_size;i++)
  if(array[i]==0)
    break;
```

Ayrıca Bakınız

Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Yaratılması ve Silinmesi

Continue Operatörü

continue operatörü, kontrolü en yakın dışsal döngüye (while, do-while veya for operatörüne), geçirir ve bir sonraki tekrarlama çağrılır. Bu operatörün amacı break operatörünün zıttıdır.

Örnek:

```
//--- Sıfır olmayan elemanların toplamı
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);
  }
```

Ayrıca Bakınız

Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Yaratılması ve Silinmesi

Nesne Oluşturma Operatörü new

new operatörü, otomatik olarak uygun büyüklükte bir nesne oluşturur, nesne yıkıcısını çağırır ve oluşturulan nesnenin bir tarifçisine dönüş yapar. Başarısızlık durumunda operatör, NULL sabiti ile karşılaştırılabilecek bir null (boş) açıklayıcıya dönüş yapar.

'new' operatörü sadece sınıf nesnelerine uygulanabilir. Yapılara uygulanamaz.

Nesne dizileri oluşturmak için kullanılamaz. Bunun için ArrayResize() fonksiyonunu kullanınız.

Örnek:

```
//+------------------------------------------------------------------+
//| Şekil oluşturma                                                  |
//+------------------------------------------------------------------+
void CTetrisField::NewShape()
  {
   m_ypos=HORZ_BORDER;
//--- rassal olarak 7 muhtemel şekil oluştur
   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;
     }
//--- çiz
   if(m_shape!=NULL)
     {
      //--- ön ayarlar
      m_shape.SetRightBorder(WIDTH_IN_PIXELS+VERT_BORDER);
      m_shape.SetYPos(m_ypos);
      m_shape.SetXPos(VERT_BORDER+SHAPE_SIZE*8);
      //--- çiz
      m_shape.Draw();
     }
//---
  }
```

Nesne tarifçisinin bellek adresine yapılan bir işaretçi olmadığı not edilmelidir.

'new' operatörü ile oluşturulmuş bir nesne, delete operatörü ile açıkça silinmelidir.

Ayrıca Bakınız

Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Yaratılması ve Silinmesi

Nesne Silme Operatörü 'delete'

'delete' operatörü new operatörü ile oluşturulmuş bir nesneyi siler. Karşılık gelen sınıfın yıkıcısını çağırır ve nesne tarafından işgal edilmiş belleği serbest bırakır. Mevcut nesnenin bir reel tarifçisi, işlenen şeklinde kullanılır. 'delete' operatörünün çalıştırılmasının ardından nesne tarifçisi geçersiz olur.

Örnek:

```
      //--- şekli sil
      delete m_shape;
      m_shape=NULL;
      //--- yeni bir şekil oluştur
      NewShape();
```

Ayrıca Bakınız

Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Yaratılması ve Silinmesi

Fonksiyonlar

Tüm görevler alt görevlere bölünebilir. Bunların her biri doğrudan bütün bir kod şeklinde gösterilmiş olabileceği gibi, daha ufak alt görevlere bölünmüş de olabilir. Bu yöntem *aşamalı geliştirme* olarak adlandırılır. Fonksiyonlar çözülecek alt görevlerin yazımında kullanılırlar. Fonksiyonun ne iş yaptığını tarif eden kod, *fonksiyon tanımı* şeklinde isimlendirilir:

```
fonksiyon_başlığı
  {
  talimatlar
  }
```

İlk çengel parantezden önceki her şey, fonksiyon tanımının *başlığını* oluşturur. Çengel parantezlerin arasında ise fonksiyon tanımının *gövdesi* yer alır. Fonksiyon başlığı dönüş değeri tipinin bir tarifini, fonksiyonun ismini (tanımlayıcı) ve biçimsel parametreleri içerir. Fonksiyona geçirilen parametrelerin sayısı sınırlıdır ve 64'ü geçemez.

Fonksiyon, programın diğer bölümlerinden gerektiği kadar çok çağrılabilir. Dönüş tipi, fonksiyon tanımlayıcısı ve parametre tipleri birlikte *fonksiyon prototipini* oluşturur.

Fonksiyon prototipi fonksiyonun bildirimidir fakat tanımı değildir. Dönüş tipinin ve argüman tiplerinden oluşan bir listenin açık bildirimi sayesinde, fonksiyon çağrıları esnasında katı tip kontrolü ve gizli tip dönüşümü mümkün olmaktadır. Fonksiyon bildirimleri kodun okunabilirliğini artırmak için sınıflarda çok sık kullanılır.

Fonksiyon tanımı fonksiyon bildirimi ile tam olarak örtüşmelidir. Bildirilen her fonksiyon tanımlanmalıdır.

Örnek:

```
double                          // dönüş değeri tipi
linfunc (double a, double b) // fonksiyon ismi ve parametre listesi
  {
                                // kompozit işlemci
   return (a + b);              // dönüş değeri
  }
```

return operatörü, içine yerleştirilen bir ifadenin değerine dönüş yapabilir. Gerektiği takdirde, ifade değeri fonksiyonun sonuç tipine dönüştürülür. Dönüş yapılabilecek şeyler şunlardır: basit tipler, basit yapılar, nesne işaretçileri. Herhangi bir diziye, sınıf nesnesine veya bileşik yapı tipli değişkenlere *return* operatörü ile dönüş yapılamaz.

Hiçbir değere dönüş yapmayan bir fonksiyon, void tipi ile bildirilmelidir.

Örnek:

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

Fonksiyona geçirilen parametreler aynı tipteki sabitlerce belirlenmiş ön tanımlı değerler alabilirler.

Örnek:

```
int somefunc(double a,
             double d=0.0001,
             int n=5,
             bool b=true,
             string s="geçirilmiş dizgi")
  {
   Print("Gereken parametre a = ",a);
   Print("Şu parametreleri geçir: d = ",d," n = ",n," b = ",b," s = ",s);
   return(0);
  }
```

Parametrelerden herhangi birinin değeri ön-tanımlı ise, bütün diğer parametrelerin de değerleri ön-tanımlı olmadır.

Hatalı bildirim Örneği:

```
int somefunc(double a,
             double d=0.0001,    // ön tanımlı değer 0.0001 bildirilmiş
             int n,              // ön tanımlı değer belirlenmemiş !
             bool b,             // ön tanımlı değer belirlenmemiş !
             string s="geçirilmiş dizgi")
  {
  }
```

Ayrıca Bakınız

Aşırı Yükleme, Sanal Fonksiyonlar, Polimorfizm

Fonksiyon Çağrısı

Daha önce tarifi yapılmamış bir isim ifade içerisinde gözüküyorsa ve bunu bir sol parantez takip ediyorsa, kavramsal açıdan bu isim fonksiyonun ismi kabul edilir.

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

Argümanlar (biçimsel parametreler) değer ile geçirilirler, yani her bir ifade x1,..., xn hesaplanır ve ardından değer, fonksiyona geçirilir. İfadelerin hesaplanma sıraları ve değerlerin yüklenme sıraları kesin değildir. Çalıştırma esnasında sistem fonksiyona geçirilen argümanların tiplerini ve sayılarını kontrol eder. Bu şekilde fonksiyona yapılan adreslemelere değer çağrısı denir.

Fonksiyon çağrısı, değeri fonksiyonun dönüş değeri olan bir ifadedir. Yukarıda tarif edilen fonksiyon tipi, dönüş değerinin tipi ile uyumlu olmalıdır. Fonksiyon global alan üzerinde, programın herhangi bir yerinde (diğer fonksiyonların dışında) bildirilebilir veya tarif edilebilir. Fonksiyon başka fonksiyonların içerisinde bildirilemez veya tarif edilemez.

Örnekler:

```
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);
  }
```

Bir fonksiyon ön tanımlı ayarlar ile çağrılırken, geçirilecek parametrelerin listesi sınırlandırılabilir buna rağmen ön tanımlı parametreler kesinlikle geçirilmelidir.

Örnekler:

```
void somefunc(double init,
              double sec=0.0001, // varsayılan değerleri ayarla
              int level=10);
//...
somefunc();                      // Yanlış çağrı. bir parametre sunulmalı.
somefunc(3.14);                  // Doğru çağrı
somefunc(3.14,0.0002);           // Doğru çağrı
somefunc(3.14,0.0002,10);        // Doğru çağrı
```

Fonksiyon çağrısı yapılırken parametreler atlanamaz, ön tanımlı değerleri olsa bile:

```
somefunc(3.14, , 10);           // Yanlış çağrı -> ikinci parametre atlandı.
```

Ayrıca Bakınız

Aşırı Yükleme, Sanal Fonksiyonlar, Polimorfizm

Parametrelerin Geçirilmesi

Makine dilinin argümanları bir alt programa (fonksiyona) geçirmesinin iki yolu vardır. İlk yol değer aracılığıyla aktarmaktır. Argüman değeri bir biçimsel fonksiyon parametresinin içine kopyalanır. Fonksiyon içinde bu parametrede yapılan değişikliklerin hiçbirinin, ilgili çağrı argümanı üzerinde bir etkisi yoktur.

```
//+------------------------------------------------------------------+
//| Parametrelerin değer ile geçirilmesi                             |
//+------------------------------------------------------------------+
double FirstMethod(int i,int j)
  {
   double res;
//---
   i*=2;
   j/=2;
   res=i+j;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int a=14,b=8;
   Print("çağrı öncesi a ve b:",a," ",b);
   double d=FirstMethod(a,b);
   Print("çağrı sonrası a ve b:",a," ",b);
  }
//--- Betiğin çalışma sonucu
//  çağrı öncesi a ve b: 14 8
//  çağrı sonrası a ve b: 14 8
```

İkinci yöntem referans ile geçirmedir. Bu durumda, fonksiyon parametresine bir değer değil, bir değişken referansı geçirilir. Bu, çağrıda belirtilen gerçek parametreyi ifade etmek için kullanılır. Yani, fonksiyonun çağrılmasında kullanılan argümanlar parametre değişimlerinden etkilenecektir.

```
//+------------------------------------------------------------------+
//| Parametrelerin referansla geçirilmesi                            |
//+------------------------------------------------------------------+
double SecondMethod(int &i,int &j)
  {
   double res;
//---
   i*=2;
   j/=2;
   res=i+j;
//---
```

```
   return(res);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int a=14,b=8;
   Print("çağrı öncesi a ve b:",a," ",b);
   double d=SecondMethod(a,b);
   Print("çağrı sonrası a ve b:",a," ",b);
  }
//+------------------------------------------------------------------+
//--- script çalışmasının sonucu
//  çağrı öncesi a ve b: 14 8
//  çağrı sonrası a ve b: 28 4
```

Yukarıda açıklanan yolların ikisi de MQL5 içinde kullanılır, bir istisna dışında: diziler, yapı tipli değişkenler ve sınıf nesneleri her zaman referans ile geçirilirler. Gerçek parametrelerdeki (fonksiyon çağrılarında geçirilen argümanlar) değişikliklerden kaçınmak amacıyla, const erişim belirtecini kullanabilirsiniz. *const* belirteci ile bildirilmiş değişkenlerin içeriğinde değişiklik yapılması denendiğinde, derleyici bir hata oluşturacaktır.

Not

Bilinmelidir ki parametreler fonksiyona ters sırada geçirilirler. Yani, ilk olarak son parametre hesaplanır ve geçirilir, ardından sondan bir önceki vb. Son olarak hesaplanıp geçirilen parametre ise, açılan parantezin yanında ilk sırada durandır.

Örnek:

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

   func(a[i],a[i++],"İlk çağrı (i = "+string(i)+")");
   func(a[i++],a[i],"İkinci çağrı (i = "+string(i)+")");
// Sonuç:
// İlk çağrı (i = 0) : par1 = 1     par2 = 0
// İkinci çağrı (i = 1) : par1 = 1     par2 = 1

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

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

İlk çağrıda (yukarıdaki örneğe bakın) ***i*** değişkeni ilk olarak dizgilerin birleştirilmesinde kullanılır:

```
  "İlk çağrı (i = "+string(i)+")"
```

Burada 'i' değeri değişmemektedir. Ardından ***i*** değişkeni ***a[i++]*** dizi elemanının hesabında kullanılır. Yani i indisli elemana ulaşıldığında, ***i*** değeri artırılır. Değiştirilmiş ***i*** değerine sahip ilk parametre, ancak bundan sonra hesaplanacaktır.

İkinci çağrıda i'nin aynı değeri (fonksiyon çağrısının ilk aşamasında hesaplanan) tüm parametrelerin hesaplanmasında kullanılır. ***i*** değişkeni sadece ilk parametreler hesaplandıktan sonra yeniden değişir.

Ayrıca Bakınız

Değişkenlerin Görünürlük Alanları ve Ömürleri, Aşırı Yükleme, Sanal Fonksiyonlar, Polimorfizm

Fonksiyonun Aşırı Yüklenmesi

Fonksiyonun ismi çoğu zaman temel amacını yansıtma eğilimi gösterir. Bir kural olarak, okunaklı programlar iyi seçilmiş çeşitli tanımlayıcılar içerir. Bazen farklı fonksiyonların aynı amaçla kullanıldıkları olur. Örneğin, çifte duyarlık sayılarını içeren bir dizinin ortalamasını hesaplayan bir fonksiyonu ve bir de aynı fonksiyonun tamsayı dizileriyle çalışan bir versiyonunu göz önüne getirelim. İkisi de AverageFromArray şeklinde isimlendirilmek için uygundur:

```
//+------------------------------------------------------------------+
//| double tipi bir dizinin ortalamasının hesaplanması               |
//+------------------------------------------------------------------+
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];    // double için toplama işlemi
     }
   aver=sum/size;       // Toplamı ortalamaya böldük
//---
   Print("double tipli dizinin ortalamasının hesaplanması");
   return aver;
  }
//+------------------------------------------------------------------+
//| int tipi bir dizinin ortalamasının hesaplanması                  |
//+------------------------------------------------------------------+
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];     // int için toplama işlemi
     }
   aver=(double)sum/size;// double tipli miktarı alıp ve böl
//---
   Print("int tipli dizinin ortalamasının hesaplanması");
   return aver;
  }
```

İki fonksiyon da Print() fonksiyonuyla verilen çıkış mesajını içermektedir;

```
   Print("int tipli dizinin ortalamasının hesaplanması");
```

Derleyici, gereken bir fonksiyonu argümanlarının tiplerine ve sayılarına göre seçer. Bu seçime referans oluşturan kural, *imza eşleştirme algoritması* olarak adlandırılır. İmza, fonksiyon bildiriminde kullanılan tiplerin bir listesidir.

Örnek:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   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);
  }
//--- Betiğin sonucu
// int tipli dizinin ortalamasının hesapla
// double tipli dizinin ortalamasının hesapla
// int_aver= 3.00000000    double_aver= 3.30000000
```

Fonksiyonun aşırı yüklenmesi, aynı isimde ama farklı parametrelerle birkaç fonksiyon oluşturma işlemidir. Bu, fonksiyonun aşırı yüklenmiş versiyonlarında, argümanların ve/veya argüman tiplerinin farklı olmaları gerektiği anlamına gelir. Belirli bir fonksiyon türevi, fonksiyon bildirimindeki parametre listesine karşılık gelen argümanların listesi temel alınarak seçilir.

Aşırı yüklenmiş bir fonksiyon çağrıldığında, derleyicinin uygun fonksiyonu seçmek için bir algoritmaya sahip olması gerekir. Bu seçimi yürüten algoritma mevcut tiplerin dönüşümüne bağlıdır. En iyi örtüşme benzersiz olmalıdır. Aşırı yüklenmiş bir fonksiyonun, tüm diğer türevler arasında, en az bir argüman için en iyi eşleşme olması gerekir. Ayrıca, fonksiyon tüm diğer argümanlar için, diğer türevlerden daha kötü olmayan bir eşleşmeye sahip olmalıdır.

Aşağıda bir eşleştirme algoritması yer almaktadır.

Aşırı Yüklenmiş bir Fonksiyonun Seçiminin Algoritması

1. Katı eşleşme kullan (eğer mümkünse).
2. Standart tip artırımını dene.
3. Standart tip dönüşümünü dene.

Standart tip artırımı diğer standart dönüşümlerden daha iyidir. Artırma float tipinden double tipine, bool, char, short veya enum tiplerinden int tipine yapılan dönüşümdür. Tamsayı tipli benzer dizilerin dönüşümleri de yine tip dönüşümü içerisinde değerlendirilir. Benzer tipler şunlardır: bool, char ve uchar, bu üç tipin hepsi tek-baytlık tamsayılardır. 'short' ve 'ushort' çift-bayt tamsayılar; int, uint ve color; long, ulong ve datetime 4-bayt tamsayılardır.

Elbette en iyisi katı eşleşmedir. Böyle bir tutarlılığı sağlamak için tip dönüşümü kullanılır. Derleyici belirsiz durumlarla başa çıkamaz. Bu yüzden, küçük tip farklılıklarına ve aşırı yüklenmiş fonksiyonu anlaşılmaz kılan gizli dönüşümlere güvenmemelisiniz.

Şüphe durumunda katı uyumu sağlayacak açık dönüşümler kullanın.

Aşırı yüklenmiş fonksiyonun MQL5 içindeki örneği ArrayInitialize() fonksiyonu sayfasında görülebilir.

Fonksiyon aşırı yükleme kuralları sınıf yöntemlerinin aşırı yüklenmesinde de uygulanır

Sistem fonksiyonlarının aşırı yüklenmesine izin verilir. Bu yüzden derleyicinin gerekli fonksiyonu doğru seçebildiği gözlenmelidir. Örneğin, MathMax() sistem fonksiyonunu 4 ayrı şekilde aşırı yükleyebiliriz ama bunlardan sadece ikisi doğru olacaktır.

Örnek:

```
// 1. aşırı yükleme izni var - fonksiyon, parametre sayısı açışından gömülü MathMax()
double MathMax(double a,double b,double c);

// 2. aşırı yükleme izni yok!
// parametre sayıları farklı ama sonuncusu bir varsayılan (ön tanımlı) değere sahip
// bu, çağrı sırasında sistem fonksiyonunun gizlenmesine yol açıyor ve bu kabul ediler
double MathMax(double a,double b,double c=DBL_MIN);

// 3. aşırı yükleme izni var - a ve b parametrelerinin tipleri aracılığıyla normal aş
double MathMax(int a,int b);

// 4. aşırı yükleme izni yok!
// parametrelerin tipleri ve sayıları orjinal double MathMax(double a,double b) ile a
int MathMax(double a,double b);
```

Ayrıca Bakınız

Aşırı Yükleme, Sanal Fonksiyonlar, Polimorfizm

İşlemin Aşırı Yüklenmesi

Kodun okunmasını ve yazılmasını kolaylaştırması için aşırı yüklemeye izin verilir. Aşırı yükleme operatörü operator anahtar sözcüğü ile yazılır. Şu operatörler aşırı yüklenebilir:

- ikili +,-,/,*,%,<<,>>,==,!=,<,>,<=,>=,=,+=,-=,/=,*=,%=,&=,|=,^=,<<=,>>=,&&,||,&,|,^
- tekli +,-,++,--,!,~
- atama operatörü =
- indisleme operatörü []

İşlemin aşırı yüklenmesi, karmaşık nesneler - yapılar ve sınıflar için (basit ifadeler şeklinde yazılmış) işlem notasyonunun kullanılmasını sağlar. Kodu aşırı yüklenmiş operatörler kullanarak yazmak kaynak kodunun görünümünü basitleştirir, çünkü daha karmaşık uygulamalar gizlenecektir.

Örneğin, gerçel ve imajiner kısımlardan meydana gelen karmaşık sayıları ele alalım. Bu sayılar matematikte sıklıkla kullanılırlar. MQL5 dili içerisinde karmaşık sayıları temsil eden bir veri tipi yer almaz ama bir yapı veya sınıf şeklinde yeni bir tip oluşturmak mümkündür. Karmaşık yapının bildirimi ve dört aritmetik işlem gerçekleştiren dört ayrı yöntemin tanımı:

```
//+------------------------------------------------------------------+
//| Karmaşık sayılı işlemler için bir yapı                           |
//+------------------------------------------------------------------+
struct complex
  {
   double            re; // Gerçel kısım
   double            im; // İmajiner kısım
   //--- Yapıcılar
                     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) { }
   //--- Aritmetik işlemler
   complex           Add(const complex &l,const complex &r) const;  // Toplama
   complex           Sub(const complex &l,const complex &r) const;  // Çıkarma
   complex           Mul(const complex &l,const complex &r) const;  // Çarpma
   complex           Div(const complex &l,const complex &r) const;  // Bölme
  };
```

Şimdi, kodumuzda karmaşık sayıları temsil eden değişkenleri bildirilebilir, ardından bunlarla çalışabiliriz.

Örneğin:

```
void OnStart()
  {
//--- Karmaşık tipli değişkenleri bildir ve başlat
   complex a(2,4),b(-4,-2);
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
//--- İki sayıyı topla
```

```
   complex z;
   z=a.Add(a,b);
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- İki sayıyı çarp
   z=a.Mul(a,b);
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- İki sayıyı böl
   z=a.Div(a,b);
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
```

Ama, alışıldık aritmetik işlemleri karmaşık sayılarla yapmak için; "+", "-", "*" ve "/" şeklindeki alışıldık operatörleri kullanmak daha uygun olacaktır.

'operator' anahtar kelimesi, dönüşümü gerçekleştiren bir üye fonksiyonunu tanımlamak için kullanılır. Sınıf nesnesi değişkenleri için yapılan tekli ve ikili işlemler, statik olmayan üye fonksiyonları gibi aşırı yüklenmiş olabilirler. Bunlar sınıf nesnesi üzerinde gizlice hareket ederler.

Bir sınıf nesnesi veya bu sınıfın bir nesnesinin işaretçisi şeklinde bir veya iki argümanı olan düzenli fonksiyonlar gibi, bir çok ikili işlem de aşırı yüklenebilir. Bizim oluşturduğumuz tip (complex) için, bildirim içindeki aşırı yükleme şöyle gözükecektir:

```
   //--- Operatörler
   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)); }
```

Betik örneğinin bütünü:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- complex tipindeki değişkenleri bildir ve başlat
   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;
//--- İki sayıyı topla
   complex z=a+b;
   PrintFormat("a+b=%.2f+i*%.2f",z.re,z.im);
//--- İki sayıyı çarp

   z=a*b;
   PrintFormat("a*b=%.2f+i*%.2f",z.re,z.im);
//--- İki sayıyı böl
```

```
   z=a/b;
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//---
  }
//+------------------------------------------------------------------+
//| Karmaşık sayılı işlemler için bir yapı                           |
//+------------------------------------------------------------------+
struct complex
  {
   double            re; // Gerçel kısım
   double            im; // İmajiner kısım
   //--- Yapıcılar
                     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) { }
   //--- Aritmetik işlemler
   complex           Add(const complex &l,const complex &r) const;  // Toplama
   complex           Sub(const complex &l,const complex &r) const;  // Çıkarma
   complex           Mul(const complex &l,const complex &r) const;  // Çarpma
   complex           Div(const complex &l,const complex &r) const;  // Bölme
   //--- İkili operatörler
   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)); }
  };
//+------------------------------------------------------------------+
//| Toplama                                                          |
//+------------------------------------------------------------------+
complex complex::Add(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re+r.re;
   res.im=l.im+r.im;
//--- Sonuç
   return res;
  }
//+------------------------------------------------------------------+
//| Çıkarma                                                          |
//+------------------------------------------------------------------+
complex complex::Sub(const complex &l,const complex &r) const
  {
   complex res;
//---
   res.re=l.re-r.re;
   res.im=l.im-r.im;
//--- Sonuç
```

```
   return res;
  }
//+------------------------------------------------------------------+
//| Çarpma                                                           |
//+------------------------------------------------------------------+
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;
//--- Sonuç
   return res;
  }
//+------------------------------------------------------------------+
//| Bölme                                                            |
//+------------------------------------------------------------------+
complex complex::Div(const complex &l,const complex &r) const
  {
//--- Boş karmaşık sayılar
   complex res(EMPTY_VALUE,EMPTY_VALUE);
//--- Sıfır için kontrol et
   if(r.re==0 && r.im==0)
     {
      Print(__FUNCTION__+": sayı sıfıra eşittir");
      return(res);
     }
//--- Yardımcı değişkenler
   double e;
   double f;
//--- Hesaplama Türevinin seçimi
   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;
     }
//--- Sonuç
   return res;
  }
```

Bir tekil sınıf nesnesini veya bunun işaretçisini argümanı olarak kabul eden normal fonksiyonlar gibi, sınıflar için gerçekleştirilen bir çok tekli işlem de aşırı yüklenmiş olabilir. "-" ve "!" tekli işlemlerinin aşırı yüklenmesini ekle.

```
//+------------------------------------------------------------------+
//| Karmaşık sayılı işlemler için bir yapı                           |
//+------------------------------------------------------------------+
struct complex
  {
   double            re;       // Gerçel kısım
   double            im;       // İmajiner kısım
...
   //--- Tekli operatörler
   complex operator-()  const; // Tekli eksi
   bool    operator!()  const; // Red
  };
...
//+------------------------------------------------------------------+
//| "Tekil eksi" operatörünün aşırı yüklenmesi                       |
//+------------------------------------------------------------------+
complex complex::operator-() const
  {
   complex res;
//---
   res.re=-re;
   res.im=-im;
//--- Sonuç
   return res;
  }
//+------------------------------------------------------------------+
//| "Mantıksal red" operatörünün aşırı yüklenmesi                    |
//+------------------------------------------------------------------+
bool complex::operator!() const
  {
//--- Karmaşık sayının gerçel ve imajiner kısımları sıfır mı?
   return (re!=0 && im!=0);
  }
```

Artık karmaşık sayıyı sıfır değeri için kontrol edebilir ve red değeri alabiliriz:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- complex tipindeki değişkenleri bildir ve başlat
   complex a(2,4),b(-4,-2);
```

```
   PrintFormat("a=%.2f+i*%.2f,   b=%.2f+i*%.2f",a.re,a.im,b.re,b.im);
//--- İki sayıyı böl
   complex z=a/b;
   PrintFormat("a/b=%.2f+i*%.2f",z.re,z.im);
//--- Varsayılan olarak bir karmaşık sayı sıfıra eşittir (Yani re==0 ve im==0)
   complex zero;
   Print("!zero=",!zero);
//--- Negatif değer ata
   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);
//--- Bir kez daha sıfır için kontrol et
   Print("!zero=",!zero);
//---
  }
```

Basit tipli yapıların birinden diğerine doğrudan kopyalanabildiği düşünüldüğünde, "=" atama operatörüne aşırı yükleme yapmamızın gerekli olmadığı not edilmelidir. Artık, karmaşık sayıları alışıldık tarzda kullanan hesaplamalar için bir kod yazabiliriz.

İndisleme operatörünün aşırı yüklenmesi, nesneye iliştirilmiş dizilerin değerlerinin basitçe alınmasını sağlar. Ayrıca kodun okunabilirliğini artırmaya da yardımcı olur. Örneğin, dizgi içinde belirli konumdaki bir sembole erişim sağlamamız gerekebilir. Dizgi, MQL5 içinde ayrı bir tiptir (string), sembollerden oluşan bir dizi değildir. Ama aşırı yüklenmiş bir indisleme operatörünün yardımıyla, CString sınıfı içerisinde basit ve saydam bir çalışma sağlanabilir:

```
//+------------------------------------------------------------------------------+
//| Bir dizgideki sembollere, sembol dizisiymiş gibi erişim sağlamak için bir sınıf |
//+------------------------------------------------------------------------------+
class CString
  {
   string            m_string;

public:
                     CString(string str=NULL):m_string(str) { }
   ushort operator[] (int x) { return(StringGetCharacter(m_string,x)); }
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Bir dizgiden sembolleri almak için bir dizi
   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;
//--- str değişkeni içindeki sembolleri kullanarak bir cümle oluştur
   for(int i=0,n=ArraySize(x);i<n;i++)
     {
```

```
      res+=ShortToString(str[x[i]]);
     }
//--- Sonucu göster
   Print(res);
  }
```

İndisleme operatörünün aşırı yüklenmesine bir örnek de matris işlemleridir. Matris iki boyutlu bir dinamik diziyi ifade eder ama dizi büyüklüğü peşin olarak tanımlanmaz. Bu yüzden, array[][] şeklinde bir diziyi, ikinci boyutun büyüklüğünü tanımlamadan ve diziyi bir parametre olarak geçirmeden bildiremezsiniz. Bu soruna getirilebilecek özel bir çözüm CMatrix sınıfıdır. Bu, CRow sınıf nesnelerinin bir dizisini içerir.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Matrislerin toplanması ve çarpımı işlemleri
   CMatrix A(3),B(3),C();
//--- Satılar için bir dizi hazırla
   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};
//--- Matrisleri doldur
   A[0]=a1; A[1]=a2; A[2]=a3;
   B[0]=b1; B[1]=b2; B[2]=b3;
//--- Matrisleri Uzmanlar günlüğüne çıktıla
   Print("---- Matris A elemanları");
   Print(A.String());
   Print("---- Matris B elemanları");
   Print(B.String());

//--- Matrislerin toplamı
   Print("---- A ve B matrislerinin toplamı");
   C=A+B;
//--- Biçimlendirilmiş dizgi ifadesini çıktıla
   Print(C.String());

//--- Matrislerin çarpımı
   Print("---- A ve B matrislerinin çarpımı");
   C=A*B;
   Print(C.String());

//--- Şimdi değerleri matrix[i][j] dinamik dizileri şeklinde nasıl elde edeceğimizi gö
   Print("C matrisi değerlerini elementsel olarak çıktıla");
//--- Matris satırlarını - CRow nesneleri - bir döngü içinde incele
   for(int i=0;i<3;i++)
     {
      string com="| ";
      //--- Değer için matris satırlarından
```

```
      for(int j=0;j<3;j++)
        {
         //--- Matris elemanlarını satır ve sütun numarası kullanarak al
         double element=C[i][j];// [i] - m_rows[] dizisi içinde CRow'a erişim,
                                // [j] - CRow içinde yapılan indislemenin aşırı yükler
         com=com+StringFormat("a(%d,%d)=%G ; ",i,j,element);
        }
       com+="|";
       //--- Satır değerlerinin çıktısını al
       Print(com);
      }
  }
//+------------------------------------------------------------------+
//| "Row" sınıfı                                                     |
//+------------------------------------------------------------------+
class CRow
  {
private:
   double            m_array[];
public:
   //--- Yapıcılar ve bir yıkıcı
                     CRow(void)           { ArrayResize(m_array,0);     }
                     CRow(const CRow &r)  { this=r;                     }
                     CRow(const double &array[]);
                    ~CRow(void){};
   //--- Satırdaki elemanların sayısı
   int               Size(void) const     { return(ArraySize(m_array));}
   //--- Değerleriyle bir dizgiye dönüş yapar
   string            String(void) const;
   //--- İndisleme operatörü
   double            operator[](int i) const  { return(m_array[i]);    }
   //--- Atama operatörleri
   void              operator=(const double  &array[]); // Bir dizi
   void              operator=(const CRow & r);         // Başka bir CRow nesnesi
   double            operator*(const CRow &o);          // Çarpım için CRow nesnesi
  };
//+------------------------------------------------------------------+
//| Tek satırlı bir dizinin başlatılması için yapıcı                 |
//+------------------------------------------------------------------+
void  CRow::CRow(const double &array[])
  {
   int size=ArraySize(array);
//--- Eğer dizi boş değilse
   if(size>0)
     {
      ArrayResize(m_array,size);
      //--- Değerler ile doldur
      for(int i=0;i<size;i++)
         m_array[i]=array[i];
```

```
      }
//---
  }
//+------------------------------------------------------------------+
//| Dizi için atama işlemi                                           |
//+------------------------------------------------------------------+
void CRow::operator=(const double &array[])
  {
   int size=ArraySize(array);
   if(size==0) return;
//--- Diziyi değerlerle doldur
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++) m_array[i]=array[i];
//---
  }
//+------------------------------------------------------------------+
//| CRow için atama operatörü                                        |
//+------------------------------------------------------------------+
void CRow::operator=(const CRow  &r)
  {
   int size=r.Size();
   if(size==0) return;
//--- Diziyi değerlerle doldur
   ArrayResize(m_array,size);
   for(int i=0;i<size;i++) m_array[i]=r[i];
//---
  }
//+------------------------------------------------------------------+
//| Başka bir satırla çarpma operatörü                               |
//+------------------------------------------------------------------+
double CRow::operator*(const CRow &o)
  {
   double res=0;
//--- Doğrulamalar
   int size=Size();
   if(size!=o.Size() || size==0)
     {
      Print(__FUNCSIG__,": İki matrisin çarpımı başarısız oldu, büyüklükler farklı");
      return(res);
     }
//--- Dizileri elementsel olarak çarp ve çarpımları ekle
   for(int i=0;i<size;i++)
      res+=m_array[i]*o[i];
//--- Sonuç
   return(res);
  }
//+------------------------------------------------------------------+
//| Biçimlendirilmiş bir dizgi ifadesine dönüş yapar                 |
//+------------------------------------------------------------------+
```

```
string CRow::String(void) const
  {
   string out="";
//--- Dizi büyüklüğü sıfırdan büyükse
   int size=ArraySize(m_array);
//--- Dizinin sadece sıfır olmayan elemanlarıyla çalışacağız
   if(size>0)
     {
      out="{";
      for(int i=0;i<size;i++)
        {
         //--- Değerleri dizgiye topla
         out+=StringFormat(" %G;",m_array[i]);
        }
      out+=" }";
     }
//--- Sonuç
   return(out);
  }
//+------------------------------------------------------------------+
//| "Matrix" sınıfı                                                  |
//+------------------------------------------------------------------+
class CMatrix
  {
private:
   CRow              m_rows[];

public:
   //--- Yapıcılar ve bir yıkıcı
                     CMatrix(void);
                     CMatrix(int rows)  { ArrayResize(m_rows,rows);             }
                    ~CMatrix(void){};
   //--- Matris büyüklüklerini al
   int               Rows()       const { return(ArraySize(m_rows));          }
   int               Cols()       const { return(Rows()>0? m_rows[0].Size():0); }
   //--- Sütun değerini CRow satırı şeklinde dönüş yapar
   CRow              GetColumnAsRow(const int col_index) const;
   //--- Matris değerli bir dizgiye dönüş yapar
   string            String(void) const;
   //--- İndisleme operatörü numarası ile bir dizgiye dönüş yapar
   CRow *operator[](int i) const        { return(GetPointer(m_rows[i]));      }
   //--- Toplama operatörü
   CMatrix           operator+(const CMatrix &m);
   //--- Çarpma operatörü
   CMatrix           operator*(const CMatrix &m);
   //--- Atama operatörü
   CMatrix          *operator=(const CMatrix &m);
  };
//+------------------------------------------------------------------+
```

```
//| Ön tanılı bir yapıcı, sıfır büyüklüğünde bir satır dizisi oluştur  |
//+------------------------------------------------------------------+
CMatrix::CMatrix(void)
  {
//--- Matristeki sıfır sayılı satırlar
   ArrayResize(m_rows,0);
//---
  }
//+------------------------------------------------------------------+
//| CRow şeklindeki sütun değerine dönüş yapar                       |
//+------------------------------------------------------------------+
CRow  CMatrix::GetColumnAsRow(const int col_index) const
  {
//--- Sütundan değerleri almak için bir değişken
   CRow row();
//--- Matristeki satırların sayısı
   int rows=Rows();
//--- satır sayısı sıfırdan büyükse, işlemi başlat
   if(rows>0)
     {
      //--- col_index indisli sütun değerlerinin alımı için bir dizi
      double array[];
      ArrayResize(array,rows);
      //--- Dizinin doldurulması
      for(int i=0;i<rows;i++)
        {
         //--- i satırı için sütun sayısını kontrol et - dizi sınırlarını aşabilir
         if(col_index>=this[i].Size())
           {
            Print(__FUNCSIG__,": Hata! Sütun numarası ",col_index,"> satır büyüklüğü "
            break; // satır, başlatılmamış nesne olacak
           }
         array[i]=this[i][col_index];
        }
      //--- Dizi değerlerine göre bir CRow satırı oluştur
      row=array;
     }
//--- Sonuç
   return(row);
  }
//+------------------------------------------------------------------+
//| İki matrisin toplanması                                          |
//+------------------------------------------------------------------+
CMatrix CMatrix::operator+(const CMatrix &m)
  {
//--- Geçirilmiş matristeki satır ve sütunların sayısı
   int cols=m.Cols();
   int rows=m.Rows();
//--- Toplama sonuçlarının alınacağı matris
```

```
   CMatrix res(rows);
//--- Matrislerin büyüklükleri eşleşmeli
   if(cols!=Cols() || rows!=Rows())
     {
      //--- Toplama imkansız
      Print(__FUNCSIG__,": Matrislerin toplanması başarısız, büyüklükleri uyuşmuyor");
      return(res);
     }
//--- Yardımcı dizi
   double arr[];
   ArrayResize(arr,cols);
//--- Toplanacak satırları incele
   for(int i=0;i<rows;i++)
     {
      //--- Matris dizgilerinin toplamlarını diziye yaz
      for(int k=0;k<cols;k++)
        {
         arr[k]=this[i][k]+m[i][k];
        }
      //--- Diziyi matris satırına yerleştir
      res[i]=arr;
     }
//--- Matrislerin toplama sonucuna dönüş yap
   return(res);
  }
//+------------------------------------------------------------------+
//| İki matrisin çarpımı                                             |
//+------------------------------------------------------------------+
CMatrix CMatrix::operator*(const CMatrix &m)
  {
//--- İlk matristeki sütunların sayısı, matrise geçirilmiş satırların sayısı
   int cols1=Cols();
   int rows2=m.Rows();
   int rows1=Rows();
   int cols2=m.Cols();
//--- Toplama sonuçlarını alacak matris
   CMatrix res(rows1);
//--- Matrisler koordine edilmeli
   if(cols1!=rows2)
     {
      //--- Çarpma işlemi imkansız
      Print(__FUNCSIG__,": Matrislerin çarpımı başarısız oldu, biçim uyumsuz "
            "- ilk matristeki sütun sayısı ikinci matrisin satır sayısına eşit olmalı"
      return(res);
     }
//--- Yardımcı dizi
   double arr[];
   ArrayResize(arr,cols1);
//--- Çarpım matrisinin satırlarını doldur
```

```
   for(int i=0;i<rows1;i++)// Satırları incele
     {
      //--- Alınan diziyi sıfırla
      ArrayInitialize(arr,0);
      //--- Satırdaki elemanları incele
      for(int k=0;k<cols1;k++)
        {
         //--- Matrisin k sütununun değerlerini CRow şeklinde al
         CRow column=m.GetColumnAsRow(k);
         //--- iki satırı skaler olarak çarp ve vektörlerin çarpım sonucunu i-inci ele
         arr[k]=this[i]*column;
        }
      //--- Matrisin i-inci satırına arr[] dizisini yerleştir
      res[i]=arr;
     }
//--- Matrislerin çarpımına dönüş yap
   return(res);
  }
//+------------------------------------------------------------------+
//| Atama işlemi                                                     |
//+------------------------------------------------------------------+
CMatrix *CMatrix::operator=(const CMatrix &m)
  {
//--- Satır sayısını bul ve ayarla
   int rows=m.Rows();
   ArrayResize(m_rows,rows);
//--- Satırlarımızı, geçirilmiş matrisin değerleri ile doldur
   for(int i=0;i<rows;i++) this[i]=m[i];
//---
   return(GetPointer(this));
  }
//+------------------------------------------------------------------+
//| Matrisin dizgi şeklinde gösterimi                                |
//+------------------------------------------------------------------+
string CMatrix::String(void) const
  {
   string out="";
   int rows=Rows();
//--- Dizgileri tek tek şekillendir
   for(int i=0;i<rows;i++)
     {
      out=out+this[i].String()+"\r\n";
     }
//--- Sonuç
   return(out);
  }
```

Ayrıca Bakınız

Aşırı Yükleme, Aritmetik İşlemler, Fonksiyonun Aşırı Yüklenmesi, Öncelik Kuralları

Dışsal Fonksiyonların Tarifi

Farklı modüller içinde tanımlanmış dışsal fonksiyonlar açık yolla tarif edilmelidir. Fonksiyon tarifi dönüş tipini, fonksiyonun ismini ve tipleriyle birlikte giriş parametrelerinin kümesini içerir. Böyle bir tarifin olmaması, derleme, kurma veya çalıştırma sırasında hatalarla karşılaşılmasına yol açar. Dışsal nesneleri tarif ederken, modülü belirten *#import* anahtar kelimesini kullanın .

Örnekler:

```
#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
```

import kodu sayesinde, dışsal DLL dosyalarından veya EX5 kütüphanelerinden çağrılan fonksiyonların tarifi kolaydır. EX5 kütüphaneleri library özelliğine sahip, derlenmiş ex5 dosyalarıdır. Sadece export şekillendiricisi ile tarif edilmiş fonksiyonlar EX5 kütüphanelerinden içe aktarılabilir.

Birlikte içe aktarılmaları durumunda, DLL ve EX5 dosyalarının (bulundukları dizinden bağımsız olarak) farklı isimlere sahip olması gerektiğini lütfen not ediniz. İçe aktarılan tüm fonksiyonlar, kütüphanenin "dosya ismine" karşılık gelen kapsam çözünürlüğüne sahip olmalıdır.

Örnek:

```
#import "kernel32.dll"
   int GetLastError();
#import "lib.ex5"
   int GetLastError();
#import

class CFoo
  {
public:
   int            GetLastError() { return(12345); }
   void           func()
     {
      Print(GetLastError());           // sınıf yönteminin çağrısı
      Print(::GetLastError());         // MQL5 fonksiyonunun çağrısı
      Print(kernel32::GetLastError()); // DLL kütüphanesi fonksiyonunun kernel32.dll
      Print(lib::GetLastError());      // EX5 kütüphane fonksiyonunun lib.ex5 dosyası
     }
  };

void OnStart()
  {
   CFoo foo;
   foo.func();
  }
```

Ayrıca Bakınız

Aşırı Yükleme, Sanal Fonksiyonlar, Polimorfizm

Fonksiyonların Dışa Aktarımı

MQL5 programı içerisinde '*export*' son şekillendiricisi ile bildirilen bir program diğer MQL5 programlarında kullanılabilir. Bu tip fonksiyonlar 'dışa aktarılabilir' olarak adlandırılır ve derlemeden sonra diğer programlarca kullanılabilir.

```
int Function() export
  {
  }
```

Bu şekillendirici, ex5 dosyası tarafından dışa aktarılan EX5 fonksiyonları tablosuna fonksiyonun eklenmesi için, derleyiciye emir verir. Sadece böyle bir şekillendiriciye sahip olan fonksiyonlar, diğer MQL5 programları tarafından erişilebilir ("görünür").

library özelliği, EX5 dosyasının bir kütüphane olacağını derleyiciye bildirir. Derleyici bunu EX5 dosyasının başlığında gösterecektir.

Dışarıya aktarılabilir olması planlanan tüm fonksiyonlar *export* şekillendiricisi ile işaretlenmelidir.

Ayrıca Bakınız

Aşırı Yükleme, Sanal Fonksiyonlar, Polimorfizm

Olay İşleyici Fonksiyonları

MQL5 dili önceden tanımlanmış olayların işlenmesini sağlar. Bu olayları işleyecek fonksiyonlar MQL5 programı içinde tanımlanmalıdır. Fonksiyon ismi, dönüş tipi, (eğer varsa) parametrelerin düzenleri ve tipleri, olay işleyici fonksiyonun tarifine sıkı sıkıya uymalıdır.

Müşteri terminalinin olay işleyicisi, herhangi bir olayın işlenmesini sağlayan fonksiyonları, dönüş değeri tipine veya parametrelerin tipine göre tanımlar. Eğer bir fonksiyon için, alt tariflere uygun olmayan başka parametreler belirlenmişse, veya bu fonksiyon için istenen başka bir dönüş tipi varsa, böyle bir fonksiyon olay işleyici olarak kullanılamaz.

OnStart

OnStart() fonksiyonu Start olayının işleyicisidir ve otomatik olarak, **sadece** çalışan **script** dosyaları için oluşturulur. Parametre içermeyen void tipinde olmalıdır:

```
void OnStart();
```

OnStart() fonksiyonu için, int dönüş tipi belirlenebilir.

OnInit

OnInit() fonksiyonu Init olayının işleyicisidir. Parametresiz şekilde, void tipinde veya int tipinde olmalıdır:

```
void OnInit();
```

Init olayı, bir Uzman Danışmanın veya bir göstergenin yüklenmesi sonrasında hemen oluşturulur (betik dosyaları için oluşturulmaz). OnInit() fonksiyonu başlatma için kullanılır. OnInit() int tipi dönüş değerine sahiptir. Sıfır harici tüm dönüş kodları başlatmanın başarısız olduğu anlamına gelir ve bu durumda REASON_INITFAILED sonlandırma sebebi kodu ile Deinit olayı oluşturulur.

Uzman Danışmanlarda giriş parametrelerinin optimize edilmesi için, dönüş kodu olarak ENUM_INIT_RETCODE sayımının kullanılması tavsiye edilir. Bu değerler, en uygun sınama temsilcilerinin seçimi de dahil olmak üzere, optimizasyon biçiminin düzenlenmesi için kullanılır. Uzman danışmanın başlatılması sırasında, sınamaya başlamadan önce TerminalInfoInteger() fonksiyonunu kullanarak, sınama temsilcisinin yapılanması ve kaynakları (çekirdek sayısı, serbest bellek miktarı, vb.) hakkında bilgi alabilirsiniz. Elde edilen bilgi doğrultusunda Uzman Danışmanın optimizasyonu esnasında ilgili sınama temsilcisini kullanıp kullanmayacağınıza karar verebilirsiniz.

ENUM_INIT_RETCODE

Tanımlayıcı	Açıklama
INIT_SUCCEEDED	Başlatma işlemi başarılı, Uzman Danışmanın sınanmasına devam edilebilir. Bu kod, boş değerle aynı şeyi ifade eder - Uzman Danışman, sınayıcı içinde başarılı şekilde başlatıldı.
INIT_FAILED	Başlatma işlemi başarısız oldu. Önemli hatalar nedeniyle, sınamayı devam ettirmenin bir anlamı yok. Örneğin: Uzman Danışmanın işini yapması istenen bir göstergenin oluşturulması başarısız oldu.

Tanımlayıcı	Açıklama
	Bu dönüş değeri sıfırdan farklı bir değer anlamına gelir - Uzman danışman sınayıcı içinde başlatılamadı.
INIT_PARAMETERS_INCORRECT	Bu değer giriş parametrelerinin yanlış ayarlandığı anlamına gelir. Bu dönüş kodunu içeren sonuç dizgisi, genel optimizasyon tablosunda kırmızı ile vurgulanır. Uzman Danışmanın verilen parametreleri için sınama gerçekleştirilmeyecek, sınama temsilcisi yeni bir görev için hazır. Bu değerin alınmasından sonra, strateji sınayıcı, yeniden deneme için bu görevi diğer sınama temsilcilerine aktarmayacaktır.
INIT_AGENT_NOT_SUITABLE	Başlatma işlemi sırasında hata yok ama bir sebepten dolayı temsilci sınama için uygun değil. Örneğin, yeterli bellek yok, OpenCL desteği yok, vb. Bu dönüşün alınmasından sonra sınama temsilcisi optimizasyon işleminin sonuna kadar yeni görev kabul etmeyecektir.

Void tipli OnInit() fonksiyonu daima başarılı başlatma işlemi anlamına gelir.

OnDeinit

OnDeinit() fonksiyonu, sonlandırma yapılırken çağrılan Deinit olay işleyicisidir. void tipi ile bildirilmelidir ve const int tipi, sonlandırma sebebi kodunu içeren bir parametreye sahip olmalıdır. Farklı bir tip bildirildiği takdirde, derleyici bir uyarı oluşturacaktır ve fonksiyon çağrılmayacaktır. Betik dosyalarında Deinit olayı oluşturulmayacağı için OnDeinit() fonksiyonu da kullanılmayacaktır.

```
void OnDeinit(const int reason);
```

'Deinit' olayı Uzman danışmanlar ve göstergeler için şu durumlarda oluşturulur:

- Yeniden başlatma işleminden önce, MQL5 programının iliştirildiği grafiğin sembolünün/periyodunun değişmesine bağlı olarak;
- Yeniden başlatma işleminden önce, giriş parametrelerinin değişmesi nedeniyle;
- MQL5 programı kaldırılmadan önce.

OnTick

NewTick olayı **sadece Uzman Danışmanlar** için, Uzman danışmanın iliştirilmiş olduğu çizelgede sembol için yeni bir tik alındığında oluşturulur. OnTick() fonksiyonunun özel göstergelerde ve scriptlerde kullanımı anlamsızdır, çünkü NewTick olayı bu programlar için oluşturulmaz.

Tik olayı sadece Uzman Danışmanlar için oluşturulur ama bu Uzman Danışmanlarda OnTick() fonksiyonunun kullanılmasının zorunlu olduğu anlamına gelmemektedir. Bunun nedeni, Uzman Danışmanlarda sadece NewTick olayının değil, aynı zamanda; Timer, BookEvent ve ChartEvent olaylarının da oluşturulmuş olmasıdır. fonksiyon void tipi ile, parametreler olmadan bildirilmelidir:

```
void OnTick();
```

OnTimer

OnTimer() fonksiyonu Timer (zamanlayıcı) olayı oluştuğunda çağrılır. Bu olay, sistem saati tarafından oluşturulur ve sadece Uzman Danışmanlar ve göstergeler içindir - scriptlerde kullanılamaz. Olayın gerçekleşme sıklığı, EventSetTimer() fonksiyonu aracılığıyla alınan olay uyarılarına abone olunduğunda ayarlanır.

EventKillTimer() fonksiyonu ile timer olayının aboneliği sonlandırılabilir. Fonksiyon void tipi ile parametreler olmadan bildirilmelidir:

```
void OnTimer();
```

EventSetTimer() fonksiyonunun, OnInit() içerisinde bir kere ve EventKillTimer() fonksiyonunun da, OnDeinit() içerisinde bir kere çağrılması tavsiye edilir.

Tüm göstergeler ve Uzman Danışmanlar kendi saatleri (timer) ile çalışırlar ve olayları sadece bu saatten alırlar MQL5 programı sonlandırıldığında, EventKillTimer() fonksiyonu kullanılmamış olsa bile saat zor kullanılarak yok edilecektir.

OnTrade

Bu fonksiyon Trade (alım-satım) olayı oluştuğunda çağrılır. Bu olay, verilmiş emirlerin, açık pozisyonların, emir ve işlem geçmişlerinin listesinde bir değişiklik görüldüğünde oluşur. Bir alım-satım aktivitesi (bekleyen emir açılması, pozisyon açılıp kapanması, durdurma ayarları, bekleyen emrin tetiklenmesi, vb.) gerçekleştiğinde, emir ve işlem geçmişi ve/veya pozisyonların ve mevcut emirlerin listesi de buna bağlı olarak değişecektir.

```
void OnTrade();
```

Böyle bir olay alındığında, kullanıcı hesap doğrulama işlemini fonksiyondan bağımsız olarak koda uygulamalıdır (alım-satım stratejisi koşullarında gerekliyse). OrderSend() fonksiyonu çağrısının başarıyla tamamlanması ve 'true' dönüş değerinin alınması, alım-satım sunucusunun emri işlem sırasına aldığını ve buna bir fiş numarası verdiğini gösterir. Sunucu bu emri işlediği anda Trade olayı oluşturulacaktır. Kullanıcı, fiş değerini OnTrade() olay işleyici içerisinde kullanarak, bu emre ne olduğunu öğrenebilir.

OnTradeTransaction

Alım-satım hesabı üzerinde bazı kesin işlemler gerçekleştirildiği zaman hesabın durumu değişir. Bu eylemler şunları kapsamaktadır:

- OrderSend ve OrderSendAsync fonksiyonları kullanılarak herhangi bir MQL5 uygulamasından yapılan alım-satım istekleri ve bunların ilerideki kullanımları;
- Terminalin grafiksel arayüzü ile yapılan alım satım istekleri ve bunların ilerideki kullanımları;
- Sunucu üzerindeki bekleyen emir ve durdurma emri aktivasyonu;
- İşlemlerin alım-satım sunucusu tarafından gerçekleştirilmesi.

Yukarıda sayılanlar eylemlerin sonucunda şu alım-satım faaliyetleri gerçekleşir:

- alım-satım isteğinin işlenmesi;
- açık emirlerin değişimi;
- emir geçmişinin değişimi;

- işlem geçmişinin değişimi;
- pozisyonların değişimi.

Örneğin, bir alım emri gönderildiğinde, önce işlenir, hesap için uygun bir alım emri oluşturulur, gerçekleştirilir ve açık emir listesinden çıkarılır, sonra emir geçmişine eklenir, uygun bir işlem geçmişe eklenir ve yeni pozisyon açılır. Tüm bu eylemler alım-satım faaliyetidir. Böyle bir faaliyetin terminale ulaşımı ise TradeTransaction olayıdır. Olay, (gerçekleştiğinde) OnTradeTransaction işleyicisini çağırır

```
void  OnTradeTransaction(
   const MqlTradeTransaction&    trans,        // alım-satım faaliyeti yapısı
   const MqlTradeRequest&        request,      // istek yapısı
   const MqlTradeResult&         result        // sonuç yapısı
   );
```

İşleyici üç parametre içerir:

- **trans** - bu parametre, hesaba uygulanan alım-satım faaliyetini tanımlayan MqlTradeTransaction yapısını alır;
- **request** - bu parametre alım-satım isteğini tanımlayan MqlTradeRequest yapısını alır;
- **result** - bu parametre, alım-satım isteğinin sonucunu tanımlayan MqlTradeResult yapısını alır.

request ve **result** şeklindeki son iki parametre sadece TRADE_TRANSACTION_REQUEST tipi faaliyet için verilerle doldurulur, faaliyet verisi **trans** değişkeninin *type* parametresinden alınabilir. Bilinmelidir ki bu durumda, **trans** değişkeninde tarif edilen alım-satım faaliyetinin gerçekleşmesinin ardından, **result** değişkeninin *request_id* alanı, **request** alım-satım talebinin tarifini içerir. Request ID, gerçekleştirilen eylemi (OrderSend veya OrderSendAsync fonksiyonlarının çağrısı) ve eylemin OnTradeTransaction()'na gönderilen sonucunu ilişkilendirir.

El yordamıyla terminalden veya kod kullanarak OrderSend()/OrderSendAsync() fonksiyonları ile gönderilen bir alım-satım isteği, alım-satım sunucusu üzerinde bir çok ardışık faaliyet oluşturabilir. Bu faaliyetlerin terminale ulaşım önceliği garanti edilmez. Bu yüzden, alım-satım algoritmanızı geliştirirken, bir grup faaliyetin bir diğerinden sonra ulaşacağını düşünmemelisiniz.

- Alım-satım faaliyetlerinin tüm tipleri ENUM_TRADE_TRANSACTION_TYPE sayımı içerisinde tarif edilmiştir.
- Bir alım-satım faaliyetini tanımlayan MqlTradeTransaction yapısı faaliyet tipine bağlı olarak değişik yollarla doldurulur. Örneğin, TRADE_TRANSACTION_REQUEST tipinde bir faaliyet için sadece tip alanı (alım satım faaliyeti tipi) analiz edilmelidir. OnTradeTransaction fonksiyonunun ikinci ve üçüncü parametreleri ise (request ve result), ilave veriler için analiz edilmelidir. Daha fazla bilgi için, bakınız "Bir Alım-Satım Faaliyetinin yapısı".
- Bir alım satım faaliyetinin tarifi, emirler, işlemler ve pozisyonlarla ilgili bütün bilgileri içermez (ör. yorumlar). OrderGet*, HistoryOrderGet*, HistoryDealGet* ve PositionGet* fonksiyonları ayrıntılı bilgileri elde etmek için kullanılmalıdır.

Alım-satım faaliyetlerinin müşteri hesabına uygulanmasından sonra, faaliyetler terminalin alım-satım faaliyetleri kuyruğuna kalıcı şekilde eklenirler. Buradan, terminale ulaşım sıralarına göre OnTradeTransaction girdi noktasına gönderilirler.

Alım-satım faaliyetleri bir Uzman Danışman aracılığı ile, OnTradeTransaction işleyicisi kullanılarak işlenirken, terminal yeni ulaşan alım-satım faaliyetlerini işlemeye devam eder. Bu şekilde, alım-satım

hesabının durumu OnTradeTransaction işlemi sırasında sürekli değişebilir. Örneğin, bir MQL5 programı yeni bir emir ekleme olayını işlerken, bu emir uygulanmış, açık emirlerin listesinden silinmiş ve geçmişe eklenmiş olabilir. Uygulama daha sonra bu olaylar hakkında bilgilendirilir.

Faaliyetler kuyruğunun uzunluğu 1024 elemandır. OnTradeTransaction fonksiyonu uzun süredir yeni bir faaliyeti işliyorsa, kuyruktaki eski faaliyetlerin yerini daha yeni olanlar alabilir.

- Genel olarak, OnTrade ve OnTradeTransaction çağrıları için kesin bir oran yoktur. Bir OnTrade çağrısı birkaç OnTradeTransaction çağrısına karşılık gelir.
- OnTrade çağrısı, uygun OnTradeTransaction çağrılarından sonra gerçekleşir.

OnTester

OnTester() fonksiyonu Tester olayının işleyicisidir. Bir Uzman Danışmanın seçilen tarih aralığında sınanmasının ardından otomatik olarak oluşturulur. Fonksiyon, double tipinde ve parametreler olmadan tanımlanmalıdır:

```
double OnTester();
```

Fonksiyon çağrısı, OnDeinit() çağrısının hemen öncesinde yapılır. Dönüş değeri ile aynı tipe sahiptir - double. OnTester() Uzman Danışmanların sınanmasında kullanılabilir. Fonksiyonun ana amacı, giriş parametrelerinin genetik optimizasyonunda Custom max kriteri olarak kullanılan kesin bir değeri hesaplamaktır.

Genetik optimizasyonda, bir jenerasyon içindeki sonuçlara azalan yapılı sıralama uygulanır. Yani optimizasyon kriteri açısından en iyi sonuçlar en yüksek değerlere sahip olanlardır (Custom max optimizasyon kriteri için OnTester fonksiyonunun dönüş değerleri alınır). Bu şekildeki bir sıralamada en kötü değerler sonda yer alır. Bu değerler daha sonra dışarı atılırlar ve yeni jenerasyonun oluşumuna katılmazlar.

OnTesterInit

OnTesterInit() fonksiyonu, strateji sınayıcıdaki Uzman Danışman optimizasyonu başlamadan önce otomatik olarak oluşturulan TesterInit olayının işleyicisidir. Fonksiyon 'void' tipi ile tanımlanmalıdır. Parametreleri yoktur:

```
void OnTesterInit();
```

Optimizasyonun başlamasıyla, OnTesterDeinit() veya OnTesterPass() işleyicisine sahip bir Uzman Danışman; sınayıcı içinde belirlenen sembol ve zaman aralığı ile oluşturulan, ayrı bir terminal grafiğine otomatik olarak yüklenir ve TesterInit olayını teslim alır. Fonksiyon, optimizasyona geçilmeden önce, daha sonraki optimizasyon sonuçlarının işlenmesi aşaması için, Uzman Danışmanlarda kullanılır.

OnTesterPass

OnTesterPass() fonksiyonu TesterPass olayının işleyicisidir. Bu olay, Uzman Danışman optimizasyonu sırasında bir çerçeve alındığında otomatik olarak oluşturulur. Fonksiyon 'void' tipi ile tanımlanmalıdır. Parametreleri yoktur:

```
void OnTesterPass();
```

OnTesterPass() işleyicili bir Uzman Danışman, sınama için belirlenen sembol/zaman aralığı değerleri ile otomatik olarak ayrı bir terminal grafiğine yüklenir ve optimizasyon sırasında bir çerçeve alındığında TesterPass olaylarını yakalar. Fonksiyon, optimizasyon sonuçlarının dinamik olarak tamamlanmasını beklemeksizin "anında" işlenmesi için kullanılır. Çerçeveler, OnTester() işleyicisindeki bir tekil geçişin bitmesinden sonra çağrılabilen FrameAdd() fonksiyonu kullanılarak eklenirler.

OnTesterDeinit

OnTesterDeinit() fonksiyonu, Uzman Danışman optimizasyonunun sonrasında otomatik olarak oluşturulan TesterDeinit olayının işleyicisidir. Fonksiyon 'void' tipi ile tanımlanmalıdır. Parametreleri yoktur:

```
void OnTesterDeinit();
```

TesterDeinit() işleyicili bir Uzman Danışman, optimizasyonun başlangıcında otomatik olarak bir grafiğe yüklenir ve tamamlanmasının ardından TesterDeinit olayını teslim alır. Fonksiyon, tüm optimizasyon sonuçlarının son işlemleri için kullanılır.

OnBookEvent

OnBookEvent() fonksiyonu BookEvent olayının işleyicisidir. BookEvent sadece Uzman Danışmanlar ve göstergeler içindir, Piyasa Derinliği değiştiğinde oluşturulur. Fonksiyon 'void' tipinde olmalıdır ve 'string' tipi bir parametre içermelidir:

```
void OnBookEvent (const string& symbol);
```

Herhangi bir sembolün 'BookEvent' olayını alabilmek için, MarketBookAdd() fonksiyonunu kullanarak, önceden bu olaylara abone olmalısınız. Belirli bir semboldeki 'BookEvent' aboneliğini sonlandırmak için MarketBookRelease() fonksiyonunu çağırın.

Diğer olaylardan farklı olarak, BookEvent olayı bir yayındır. Yani bir Uzman Danışman, 'MarketBookAdd' fonksiyonunu kullanarak 'BookEvent' olaylarına abone olursa, OnBookEvent() işleyicisine sahip tüm diğer Uzman Danışmanlar da bu olayı alacaklardır. Bu yüzden, işleyiciye '*const string& symbol*' parametresiyle geçirilen sembol ismi iyi incelenmelidir.

OnChartEvent

OnChartEvent() fonksiyonu, ChartEvent olaylarından oluşan bir grubun işleyicisidir. Bu olaylar şöyle tarif edilebilir:

- CHARTEVENT_KEYDOWN — çizelge penceresinin odaklanmasındaki tuş darbesi olayı ;
- CHARTEVENT_MOUSE_MOVE — fare hareketi olayı ve fare tıklama olayı (eğer çizelge için CHART_EVENT_MOUSE_MOVE=true şeklinde ayar yapılmışsa);
- CHARTEVENT_OBJECT_CREATE — grafiksel nesne oluşturma olayı (eğer çizelge için CHART_EVENT_OBJECT_CREATE=true şeklinde ayar yapılmışsa);
- CHARTEVENT_OBJECT_CHANGE — özellikler iletişim kutusu ile bir nesnenin özelliğini değiştirme olayı;
- CHARTEVENT_OBJECT_DELETE — grafiksel nesne silinme olayı (eğer çizelge için CHART_EVENT_OBJECT_DELETE=true şeklinde ayar yapılmışsa);
- CHARTEVENT_CLICK — çizelge üzerinde fare tıklaması olayı;
- CHARTEVENT_OBJECT_CLICK — çizelge üzerindeki grafiksel nesnenin fare ile tıklanması olayı;

- CHARTEVENT_OBJECT_DRAG — fare kullanılarak grafiksel nesnenin taşınması olayı;
- CHARTEVENT_OBJECT_ENDEDIT — LabelEdit grafiksel nesnesinin giriş kutusundaki metin düzenleme işleminin bitmesi olayı;
- CHARTEVENT_CHART_CHANGE — çizelge değişimleri olayı;
- CHARTEVENT_CUSTOM+n — kullanıcı olayı kimliği (n, 0 ile 65535 arası bir değer alır).
- CHARTEVENT_CUSTOM_LAST — en son kabul edilebilir kullanıcı olayı kimliği (CHARTEVENT_CUSTOM +65535).

Funksiyon sadece Uzman Danışmanlar ve göstergeler tarafından çağrılabilir. Fonksiyon void tipinde ve 4 parametreli olmalıdır:

```
void OnChartEvent(const int id,          // Olay kimliği
                  const long& lparam,    // long tipi olay parametresi
                  const double& dparam, // double tipi olay parametresi
                  const string& sparam  // string tipi olay parametresi
  );
```

Her olay tipi için, OnChartEvent() fonksiyonunun giriş parametreleri, mevcut olayı işleyebilmek için kesin değerlere sahiptir. Bu parametreler aracılığıyla geçilen olay ve değerler aşağıdaki tabloda listelenmiştir.

Olay	id parametresinin değeri	lparam parametresinin değeri	dparam parametresinin değeri	sparam parametresinin değeri
Tuş darbesi olayı	CHARTEVENT_KEYDOWN	basılmış bir tuşun kodu	Tekrar sayısı (kullanıcının tuşa basılı tutması sonucu tekrarlanan tuş darbelerinin sayısı)	Klavye tuşlarının durumunu tarif eden bir bit maskesinin dizgi değeri
Fare olayları (Eğer çizelge için CHART_EVENT_MOUSE_MOVE=true şeklinde ayar yapılmışsa)	CHARTEVENT_MOUSE_MOVE	X koordinatı	Y koordinatı	Fare tuşlarının durumunu tarif eden bir bit maskesinin string tipli değeri
Grafiksel nesne oluşturma olayı (çizelge için CHART_EVENT_OBJECT_CREATE=true şeklinde ayar yapılmışsa)	CHARTEVENT_OBJECT_CREATE	—	—	Oluşturulan grafiksel nesnenin adı
Özellikler iletişim kutusu ile bir nesnenin özelliğinin	CHARTEVENT_OBJECT_CHANGE	—	—	Değiştirilen grafiksel nesnenin adı

Olay	id parametresinin değeri	lparam parametresinin değeri	dparam parametresinin değeri	sparam parametresinin değeri
değiştirilmesi olayı				
Grafiksel nesnenin silinmesi olayı (Eğer çizelge için CHART_EVENT_OBJECT_DELETE=true şeklinde ayar yapılmışsa)	CHARTEVENT_OBJECT_DELETE	—	—	Silinen grafiksel nesnenin ismi
Çizelge üzerinde fare tıklaması olayı	CHARTEVENT_CLICK	X koordinatı	Y koordinatı	—
Çizelge üzerindeki bir grafiksel nesnenin fare ile tıklanması olayı	CHARTEVENT_OBJECT_CLICK	X koordinatı	Y koordinatı	Olayın gerçekleştiği grafiksel nesnenin ismi
Grafiksel nesnenin fare ile sürüklenmesi olayı	CHARTEVENT_OBJECT_DRAG	—	—	Taşınan grafiksel nesnenin adı
LabelEdit grafiksel nesnesinin giriş kutusunda yapılan metin düzenleme işinin bitmesi olayı	CHARTEVENT_OBJECT_ENDEDIT	—	—	Metin düzenlemesi tamamlanan LabelEdit grafiksel nesnesinin adı
Çizelge değişimleri olayı	CHARTEVENT_CHART_CHANGE	—	—	—
N sayısının altındaki kullanıcı olayının kimliği	CHARTEVENT_CUSTOM+N	EventChartCustom() fonksiyonu tarafından ayarlanmış değer	EventChartCustom() fonksiyonu tarafından ayarlanmış değer	EventChartCustom() fonksiyonu tarafından ayarlanmış değer

OnCalculate

OnCalculate() fonksiyonu sadece özel göstergeler içinde, gösterge değerlerinin Calculate olayı tarafından hesaplanması gerektiğinde çağrılır. Bu genellikle, göstergenin hesaplanacağı sembol için yeni bir tik alınmasıyla gerçekleşir. Göstergenin bu sembol için oluşturulmuş herhangi bir fiyat çizelgesine iliştirilmesi gerekmez.

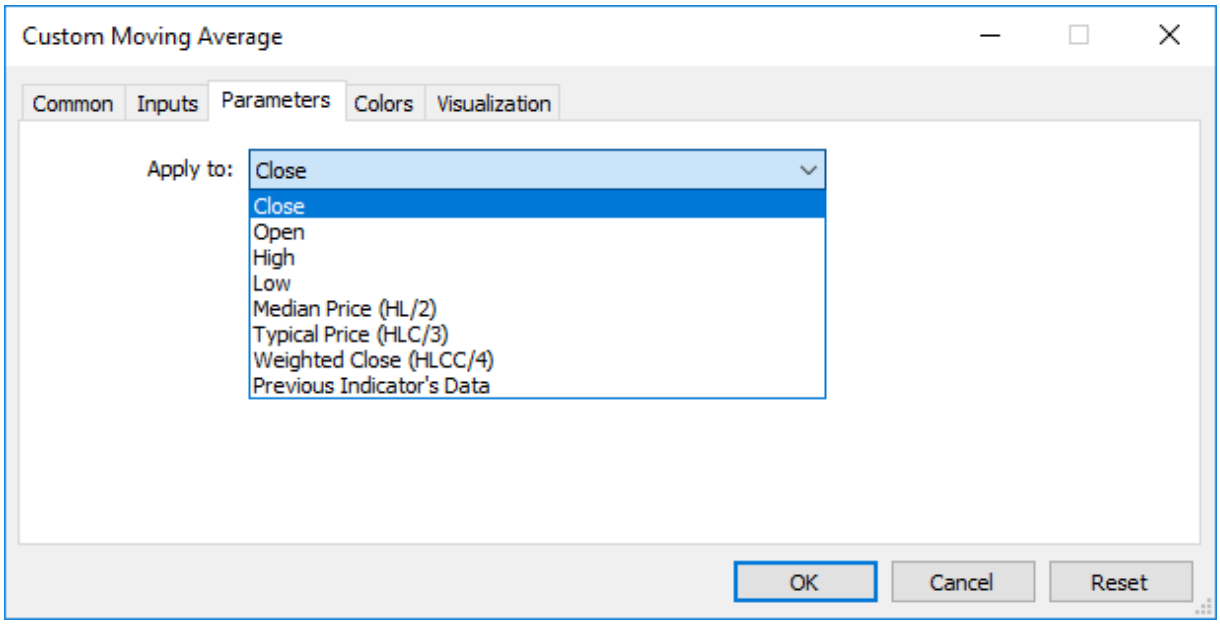
OnCalculate() fonksiyonu 'int' tipli bir dönüş değerine sahip olmalıdır. Fonksiyonun tanımının iki türü mevcuttur. Gösterge içerisinde iki versiyon birlikte kullanılamaz.

İlk versiyon, tek bir veri tamponu üzerinde hesaplanabilecek göstergeler içindir. BU tip göstergeler için Custom Moving Average örnek verilebilir.

```
int OnCalculate (const int rates_total,      // price[] dizisinin büyüklüğü
                 const int prev_calculated,  // önceki çağrıda işlenmiş çubuklar
                 const int begin,            // anlamlı verilerin başladığı yer
                 const double& price[]       // hesaplanacak dizi
   );
```

Bir zaman serisi veya başka bir göstergenin hesaplanmış bir tamponu, price[] dizisi olarak kullanılabilir. 'price[]' dizisinin indisleme yönünü belirlemek için ArrayGetAsSeries() fonksiyonunu çağırın. Çalışılmak istenen dizilerde ön tanımlı değerlere bağlı kalmamak amacıyla, ArraySetAsSeries() fonksiyonunu koşulsuz olarak çağırmalısınız.

price[] dizisi olarak kullanılacak zaman serisi veya gösterge, kullanıcı tarafından göstergenin başlatımı sırasında parametreler sekmesinden seçilebilir. Bunun için, gerekli parçayı "Uygula" alanındaki çekme liste içinde belirtmelisiniz.



Diğer MQL5 programlarından bir özel göstergenin değerlerini alabilmek için iCustom() fonksiyonu kullanılır. Bu fonksiyon sonraki işlemler için gösterge tanıtıcı değerine dönüş yapar. Uygun price[] dizisini veya başka bir göstergenin tanıtıcı değerini de ayrıca belirleyebilirsiniz. Bu parametre, özel göstergenin giriş değişkenleri listesi içinde son sırada olmalıdır.
Örnek:

```
void OnStart()
  {
//---
   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("Açılamıyor veya EX5 dosyası değil '"+terminal_path+"\\MQL5\\Indicators\\'
  }
```

Örnekte, geçirilmiş son parametre (ENUM_APPLIED_PRICE sayımından) PRICE_TYPICAL değeridir. Bu değer, özel göstergenin (High+Low+Çlose)/3 şeklinde elde edilen tipik fiyatlar üzerine inşa edileceğini belirtir. Parametre belirtilmemesi durumunda, gösterge PRICE_CLOSE değerlerinin, yani çubukların kapanış değerlerinin üzerine inşa edilir.

Gösterge tanıtıcı değerinin, price[] dizisini belirlemek amacıyla son parametre olarak geçirilmesine bir örnek de, iCustom() fonksiyonunun tarifinde verilmiştir.

İkinci versiyon, hesaplamalarda birden fazla zaman serisinin kullanıldığı diğer tüm göstergeler için düşünülmüştür.

```
int OnCalculate (const int rates_total,      // girdi zaman serilerinin büyüklüğü
                 const int prev_calculated,  // bir önceki çağrıda işlenmiş çubuklar
                 const datetime& time[],     // Zaman
                 const double& open[],       // Açılış
                 const double& high[],       // Yüksek
                 const double& low[],        // Düşük
                 const double& close[],      // Kapanış
                 const long& tick_volume[],  // Tik Hacmi
                 const long& volume[],       // Reel Hacim
                 const int& spread[]         // Makas (Alım-satım Farkı)
   );
```

open[], high[], low[] ve close[] parametreleri; açılış fiyatları, yüksek ve düşük fiyatlar ve kapanış fiyatları dizilerini içermektedir. time[] parametresi açılış zamanlarının olduğu bir diziyi içerir. spread[] parametresi (alım-satımı yapılan menkul değer için verilmişse) makas değerlerinin geçmişini içerir. volume[] ve tick_volume[] parametreleri ise, sırasıyla alım-satım ve tik hacimlerinin geçmişini içerir.

time[], open[], high[], low[], close[], tick_volume[], volume[] ve spread[] dizilerinin indisleme yönünü belirlemek için ArrayGetAsSeries() çağrısı yapabilirsiniz. Çalışılmak istenen dizilerde ön tanımlı değerlere bağlı kalmamak amacıyla ArraySetAsSeries() fonksiyonunu koşulsuz olarak çağırmalısınız.

İlk sıradaki 'rates_total' parametresi gösterge hesabı için kullanılabilecek mevcut çubukların sayısını içerir. Bu, çizelge içinde mevcut olan çubuk sayısına karşılık gelir.

OnCalculate() fonksiyonunun dönüş değeri ile ikinci giriş parametresi 'prev_calculated' arasındaki ilişki not edilmelidir. Fonksiyon çağrısı sırasında 'prev_calculated' parametresi **önceki** çağrıda OnCalculate() tarafından **dönülmüş** değeri içerir. Böylece, göstergenin hesaplanması sırasında, fonksiyonun önceki çalışmasından bu yana değişmemiş çubuklar için yapılan hesapların tekrarından kaçınılabilir.

Bu amaçla, mevcut fonksiyon çağrısındaki çubukların sayısını içeren 'rates_total' parametresinin, dönüş değeri olarak kullanılması, genellikle yeterli olacaktır. OnCalculate()'in son çağrısından bu yana, fiyat verisinde bir değişiklik gerçekleşmişse (daha derin bir geçmiş indirilmişse veya geçmiş boşlukları doldurulmuşsa), 'prev_calculated' parametresinin değeri terminal tarafından sıfırlanır.

Not: OnCalculate sıfıra dönüş yaparsa, gösterge verileri müşteri terminalinin Veri Penceresinde gözükmez.

Bunu daha iyi anlayabilmek için aşağıda verilen göstergenin çalıştırılması yararlı olacaktır.

Gösterge örneği:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- plot Line
#property indicator_label1  "Line"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDarkBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- gösterge tamponları
double         LineBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- Mevcut sembol için mevcut zaman aralığındaki çubukların sayısını al
   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]);
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

Ayrıca Bakınız

Programların Çalıştırılması, Müşteri Terminali Olayları, Olaylarla Çalışma

Değişkenler

Değişkenlerin Bildirimi

Değişkenler kullanılmadan önce bildirilmelidir. Tanımlama için benzersiz isimler kullanılır. Bir değişkeni bildirmek için tipini ve benzersiz ismini belirtmelisiniz. Değişken bildirimi bir işlemci değildir.

Bildirimde kullanılabilecek basit tipler şunlardır:

- char, short, int, long, uchar, ushort, uint, ulong - tamsayılar;
- color - RGB renklerinin tamsayı temsili;
- datetime - tarih ve zaman, 1 Ocak 1970 tarihinin sıfır anından beri geçen saniyeleri içeren işaretsiz bir tamsayı;
- bool - mantıksal değerler *true* (doğru) ve *false* (yanlış);
- double - kayan noktalı çift duyarlık sayısı;
- float - tek duyarlık kayan noktalı sayı;
- string - karakter dizgileri.

Örnekler:

```
string szInfoBox;
int    nOrders;
double dSymbolPrice;
bool   bLog;
datetime tBegin_Data   = D'2004.01.01 00:00';
color    cModify_Color = C'0x44,0xB9,0xE6';
```

Karmaşık yada bileşik tipler:

Yapılar kompozit veri tipleridir, diğer tipler kullanılarak inşa edilirler.

```
struct MyTime
  {
   int hour;    // 0-23
   int minute;  // 0-59
   int second;  // 0-59
  };
...
MyTime strTime; // Daha önce bildirilen yapının değişkeni MyTime
```

Yapıyı bildirmeden yapının değişkenlerini bildiremezsiniz.

Diziler

Dizi, aynı tipli verilerin indislenmiş dizilimidir:

```
int    a[50];       // 50 tamsayıdan oluşan tek boyutlu dizi.
double m[7][50];    // Yedi diziden oluşan iki boyutlu dizi,
                    // her biri 50 sayı içeriyor.
MyTime t[100];      // MyTime gibi elemanlar içeren dizi
```

Dizi indisleri tamsayı olmak zorundadır. Dört boyuttan yüksek dizilere izin verilmez. Dizi elemanları 0 'dan başlanarak indislenir. Tek boyutlu bir dizinin son elemanının indisi, dizinin toplam eleman sayısından bir küçüktür. Yani, 50 elemanlı bir dizinin son elemanının çağrısı a[49] şeklinde olacaktır. Aynı durum çok boyutlu dizilerde de geçerlidir: Her boyut, 0'dan 'boyut sayısı - 1' değerine kadar indislenir. Örnekteki iki boyutlu dizinin son elemanı m[6][49] şeklinde olacaktır.

Statik diziler zaman serisi şeklinde ifade edilemezler. Yani dizi elemanlarına erişimi sondan başa doğru ayarlayan ArraySetAsSeries() bunlara uygulanamaz. Eğer bir diziye zaman serilerinde olduğu gibi erişim sağlamak istiyorsanız, bir dinamik dizi nesnesi kullanın.

Dizi kapsamının dışına bir erişim denemesi olduğunda, çalıştırma alt sistemi bir kritik hata oluşturacaktır ve program duracaktır.

Dizilerle çalışmak için yerleşik metotlar

Dizilerle çalışmak için yerleşik metotların yanı sıra dizi fonksiyonları bölümündeki fonksiyonları da kullanabilirsiniz.

Metot	Analog	Açıklama
void array.Fill(const scalar value, const int start_pos=0, const int count=-1);	ArrayFill, ArrayInitialize	Diziyi belirtilen değerle doldurur
void array.Free();	ArrayFree	Dinamik dizi arabelleğini serbest bırakır ve sıfır boyutunun büyüklüğünü 0 (sıfır) olarak ayarlar
int array.Resize(const int range0_size, const int reserve); int array.Resize(const int range_sizes[], const int reserve);	ArrayResize	Dizinin birinci boyutu için yeni büyüklük ayarlar
int array.Print();	ArrayPrint	Basit türde dizinin değerlerini günlükte görüntüler
int array.Size(const int range=-1);	ArraySize, ArrayRange	Tüm dizideki (range=-1) veya belirtilen dizi boyutundaki öğelerin sayısını geri döndürür
bool array.IsDynamic();	ArrayIsDynamic	Dizinin dinamik olup olmadığını kontrol eder
bool array.IsIndicatorBuffer();		Dizinin bir gösterge arabelleği olup olmadığını kontrol eder
bool array.IsSeries();	ArrayIsSeries	Dizinin bir zaman serisi olup olmadığını kontrol eder
bool array.AsSeries();	ArrayGetAsSeries	Dizinin indeksleme yönünü kontrol eder
bool array.AsSeries(const bool as_series);	ArraySetAsSeries	Dizinin indeksleme yönünü ayarlar

Metot	Analog	Açıklama
int array.Copy(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayCopy	Dizi değerlerini başka bir diziye kopyalar
int array.Compare(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayCompare	İki basit türde dizinin veya özel yapının karşılaştırılmasının sonucunu geri döndürür
int array.Insert(const src_array[], const int dst_start, const int src_start, const int cnt);	ArrayInsert	Belirtilen indeksten başlayarak, kaynak diziden belirtilen sayıda öğeyi alıcı diziye ekler
int array.Remove(const int start_pos, const int count);	ArrayRemove	Belirtilen indeksten başlayarak, belirtilen sayıda öğeyi diziden kaldırır
int array.Reverse(const int start_pos, const int count);	ArrayReverse	Belirtilen indeksten başlayarak, belirtilen sayıda öğeyi dizide tersine çevirir
bool array.Swap(array& arr[]);	ArraySwap	İçeriği aynı türden başka bir dinamik diziyle değiştirir
void array.Sort(sort_function);	ArraySort	Sayısal dizileri birinci boyuta göre sıralar
int array.Search(scalar value, search_function);	ArrayBsearch	Dizinin birinci boyutunda bulunan ilk öğenin indeksini geri döndürür
int array.Find((scalar value, search_function);		İletilen fonksiyonu kullanarak dizide arama yapar ve bulunan ilk öğenin indeksini geri döndürür
array array.Select(scalar value, search_function);		İletilen fonksiyonu kullanarak dizide arama yapar ve bulunan tüm öğeleri içeren diziyi geri döndürür

Erişim Belirteçleri

Erişim belirteçleri derleyicinin değişkenlere, yapı veya sınıf üyelerine nasıl erişebileceğini tanımlar.

const belirteci bir değişkeni sabit olarak bildirir ve çalışma zamanı boyunca değişmesine izin vermez. Bildirildiği zaman değişkenin sadece bir sefer başlatılmasına izin verir.

Örnek:

```
int OnCalculate (const int rates_total,      // price[] dizisinin büyüklüğü
                 const int prev_calculated,  // önceki çağrıda işlenmiş çubuklar
                 const int begin,            // anlamlı verilerin başladığı yer
                 const double& price[]       // hesaplanacak dizi
   );
```

Yapı ve sınıf üyelerine erişim için şu erişim belirteçlerini kullanabilirsiniz:

- public - değişkene veya sınıf yöntemine sınırsız erişim sağlar
- protected - bu sınıfın yöntemlerinden ve genele açık olarak kalıtımlanmış sınıfların yöntemlerinden yapılan erişime izin verir. Başka türlü erişim imkansızdır;
- private - sınıf değişkenlerine ve yöntemlerine dışarıdan erişimi engeller.
- virtual - sadece sınıf yöntemlerine uygulanır (yapı yöntemlerine değil) ve derleyiciye bu yöntemin, sınıfın sanal fonksiyonlar tablosunda yer alması gerektiğini anlatır.

Bellek Sınıfları

Üç adet bellek sınıfı bulunmaktadır: static, input ve extern. Bu şekillendiriciler, karşılık gelen değişkenlerin global havuz denilen, önceden tahsis edilmiş bellek alanında dağıtıldığını derleyiciye açık olarak gösterirler. Ayrıca, değişken verilerinin özel işleniş şekillerini de belirtirler. Yerel düzeyde bildirilmiş bir değişken statik değilse, gereken bellek program yığınında otomatik olarak tahsis edilir. Statik olmayan bir dizi için ayrılmış bellek, dizinin bildirildiği bloğun görünürlük alanı dışına çıkıldığında otomatik olarak serbest bırakılır.

Ayrıca Bakınız

Veri Tipleri, Tiplerin Kapsüllenmesi ve Genişletilebilirliği, Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Oluşturulması ve Silinmesi, Bir Sınıfın Statik Elemanları

Yerel Değişkenler

Bir fonksiyon içinde bildirilen değişkenlere yerel değişken denir. Bunların etki alanları, içinde bildirildikleri fonksiyonun kapsamıyla sınırlıdır. Herhangi bir ifadenin sonucuyla başlatılabilirler. Her fonksiyon çağrısı bir yerel değişkeni başlatır. Yerel değişkenler karşılık gelen fonksiyonun bellek alanında muhafaza edilirler.

Örnek:

```
int somefunc()
  {
   int ret_code=0;
  ...
   return(ret_code);
  }
```

Değişkenin kapsamı (etki alanı), içinde ifade edildiği programın bir bölümüdür. Bir blok içinde (içsel düzeyde) bildirilmiş değişkenler, kapsam olarak bu bloğun alanına sahiptirler. Blok kapsamı değişken bildirimiyle başlar ve son sağ küme parantezi ile biter.

Yerel değişken sıfatındaki fonksiyon parametreleri gibi, fonksiyon başında bildirilmiş yerel değişkenler de blok kapsamına sahiptir. Her blok bir veya daha çok değişken bildirimi içerebilir. Bloklar iç içe geçmişse ve dış bloktaki tanımlayıcı iç bloktakiyle aynı ismi taşıyorsa, iç bloğun işlemi bitene kadar dış bloğun tanımlayıcısı gizlenir.

Örnek:

```
void OnStart()
  {
//---
   int i=5;       // fonksiyonun yerel değişkenleri
     {
      int i=10;  // fonksiyon değişkeni
      Print("Blok içi i = ",i); // sonuç i=10;
     }
   Print("Blok dışı i = ",i);  // sonuç i=5;
  }
```

Yani, iç blok, çalışırken aynı isimli dış blok tanıtıcılarının değerlerini değil, sadece kendi yerel tanıtıcılarının değerlerini görecektir.

Örnek:

```
void OnStart()
  {
//---
   int i=5;       // fonksiyonun yerel değişkenleri
   for(int i=0;i<3;i++)
      Print("Blok içi i = ",i);
   Print("Blok dışı i = ",i);
  }
/* Uygulama sonuçları
```

```
 i = 0 için içeride
 i = 1 için içeride
i = 2 için içeride
Blok dışı i = 5
*/
```

static tanımlayıcısı ile bildirilen yerel değişkenler -programın başından beri mevcut olmalarına karşın- sadece mevcut bloğun kapsamına sahiptirler.

Yığın

Her MQL5 programında, otomatik olarak oluşturulan yerel fonksiyon değişkenlerinin depolanması için yığın olarak adlandırılan özel bir bellek alanı tahsis edilir. Tüm fonksiyonlar için tek bir yığın tahsis edilir, bu yığının göstergeler için toplam boyutu 1Mb 'dır. Uzman Danışmanlarda ve betiklerde varsayılan yığın boyutu 8Mb 'dır ama yığın boyutu #property stacksize derleyici direktifi ile yeniden ayarlanabilir (bu direktif yığın boyutunu bayt cinsinden belirtir).

Statik yerel değişkenler, diğer statik ve global değişkenlerin saklandığı yerde saklanırlar (yığından ayrı olarak var olan özel bir bellek alanı içinde). Dinamik olarak oluşturulmuş değişkenler de yığından farklı bir bellek alanı kullanırlar.

Her fonksiyon çağrısıyla, statik olmayan içsel değişkenler için yığın üzerinde bir yer tahsis edilir. Fonksiyondan çıkılmasıyla birlikte bellek yeniden kullanım için uygun olacaktır.

Eğer ilk fonksiyon içinden ikincisi çağrılmışsa, ikinci fonksiyon kalan yığın alanından kendi değişkenleri için gereken alanı işgal edecektir. Yani, iç içe eklenmiş fonksiyonlar kullanılırken, yığın alanı sırasıyla her bir fonksiyon için işgal edilecektir. Bu, fonksiyonlardan birinin çağrısı sırasında hafıza sıkıntısına yol açabilir. Bu durum yığın aşımı olarak adlandırılır.

Yani, büyük yerel veriler için dinamik bellek kullanmanız daha iyi olabilir. Fonksiyon içinde yerel ihtiyaçlar için new veya ArrayResize() kullanarak gereken belleği tahsis edin. Fonksiyondan çıktığınızda delete veya ArrayFree() kullanarak belleği boşaltın.

Ayrıca Bakınız

Veri Tipleri, Tiplerin Kapsüllenmesi ve Genişletilebilirliği, Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Oluşturulması ve Silinmesi

Biçimsel Parametreler

Fonksiyona geçirilen tüm parametreler yereldir. Kapsam, fonksiyon bloğunun alanı kadardır. Biçimsel parametrelerin isimleri, dışsal değişkenlerin isimlerinden ve fonksiyon içinde tanımlanmış yerel değişkenlerin isimlerinden farklı olmalıdır. Fonksiyon bloğundaki biçimsel parametrelere aynı değerler atanabilir. Eğer bir biçimsel parametre const şekillendiricisi ile bildirilmişse, değeri fonksiyon içerisinde değiştirilemez.

Örnek:

```
void func(const int & x[], double y, bool z)
  {
   if(y>0.0 && !z)
      Print(x[0]);
  ...
  }
```

Biçimsel parametreler, sabitler kullanılarak başlatılabilir. Bu durumda başlatma değeri varsayılan olarak kabul edilir. Başlatılmış olandan sonra gelen parametreler de başlatılmalıdır.

Örnek:

```
void func(int x, double y = 0.0, bool z = true)
  {
  ...
  }
```

Böyle bir fonksiyonu çağırırken başlatılan parametreler atlanabilir, ön tanımlı olanlar bunların yerine ikame edilecektir.

Örnek:

```
func(123, 0.5);
```

Basit tipli parametreler 'değer ile' geçirilir. Yani çağrılan fonksiyonun içinde, söz konusu parametreye karşılık gelen aynı tipli yerel değişken üzerinde yapılan oynamalar, çağıran fonksiyona yansımaz. Yapı verilerini veya tiplerini içeren diziler her zaman 'referans ile' geçirilirler. Dizi veya yapı içeriklerinin değiştirilmesi istenmiyorsa, bu tipteki parametreler *const* anahtar sözcüğü ile bildirilmelidir.

Basit tipli parametreleri referans ile geçirebilirsiniz. Ama, çağıran fonksiyon içinde ilgili parametrelerin değiştirilmesi durumunda, karşılık gelen referans ile geçirilmiş değişkenler de etkilenecektir. Bir parametrenin referans ile geçirildiğini belirtmek amacıyla veri tipinden sonra & şekillendiricisini kullanın.

Örnek:

```
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;
  }
```

Referans ile geçirilmiş parametreler ön-tanımlı değerler ile başlatılamaz.

Bir fonksiyona en fazla 64 parametre geçirilebilir.

Ayrıca Bakınız

Girdi Değişkenler, Veri tipleri, Tiplerin Kapsüllenmesi ve Genişletilebilirliği, Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Oluşturulması ve Silinmesi

Statik Değişkenler

Bellek sınıfı static, bir statik değişkeni tanımlar ve veri tipinden önce gösterilir.

Örnek:

```
int somefunc()
  {
   static int flag=10;
  ...
   return(flag);
  }
```

Statik değişkenler, herhangi bir ifade ile başlatılabilen basit yerel değişkenlerin aksine, bir sabitle veya veri tipine karşılık gelen bir ifadeyle başlatılabilirler.

Statik değişkenler programın çalıştırılmasıyla var olurlar ve özelleştirilmiş OnInit() fonksiyonunun başlatılmasından önce, sadece bir kez başlatılırlar. Eğer başlangıç değeri belirtilmemişse, statik bellek sınıfının değişkenleri sıfır değeri ile başlatılır.

static anahtar sözcüğüyle bildirilmiş yerel değişkenler, değerlerini fonksiyon ömrü boyunca korurlar. Sonraki her fonksiyon çağrısında, bir önceki çağrıda sahip oldukları değerleri korurlar.

Fonksiyonun biçimsel parametreleri hariç, blok içindeki tüm değişkenler statik olarak tanımlanabilir. Eğer yerel düzeyde bildirilen bir değişken statik değilse, bu değişken için gereken bellek program yığınında otomatik olarak tahsis edilir.

Örnek:

```
int Counter()
  {
   static int count;
   count++;
   if(count%100==0) Print("Fonksiyon sayacı ",count," defa çağrıldı");
   return count;
  }
void OnStart()
  {
//---
   int c=345;
   for(int i=0;i<1000;i++)
     {
      int c=Counter();
     }
   Print("c =",c);
  }
```

Ayrıca Bakınız

Veri tipleri, Tiplerin Kapsüllenmesi ve Genişletilebilirliği, Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Oluşturulması ve Silinmesi, Statik Sınıf Üyeleri

Global Değişkenler

Global değişkenler, fonksiyon tarifinin dışında bildirilen değişkenlerdir. Fonksiyonlarla aynı düzeyde tanımlanırlar, yani hiçbir blok içinde yerel değildirler.

Örnek:

```
int GlobalFlag=10;   // Global değişken
int OnStart()
  {
  ...
  }
```

Global değişkenlerin kapsamı tüm programdır. Program içinde tanımlanmış tüm fonksiyonlar için erişilebilir durumdadırlar. Başka bir başlatma değeri açık yolla belirtilmemişse, başlangıç değerleri sıfırdır. Sadece bir sabitle veya onun tipine karşılık gelen sabit bir ifade ile başlatılabilirler.

Global değişkenler, programın müşteri terminali hafızasına yüklenmesinin hemen ardından ardından sadece bir kere ve Init olayı işlenmeden önce başlatılırlar. Sınıf nesnelerini temsil eden global değişkenler başlatılırken, ilgili sınıfın yapıcı fonksiyonu çağrılır. Betik dosyalarında Start olayı işlenmeden önce başlatılırlar.

Not: Global düzeyde bildirilen değişkenler, GlobalVariable...() fonksiyonu kullanılarak erişilebilen müşteri terminalinin global değişkenleri ile karıştırılmamalıdır.

Ayrıca Bakınız

Veri Tipleri, Tiplerin Kapsüllenmesi ve Genişletilebilirliği, Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Oluşturulması ve Silinmesi

Girdi Değişkenleri

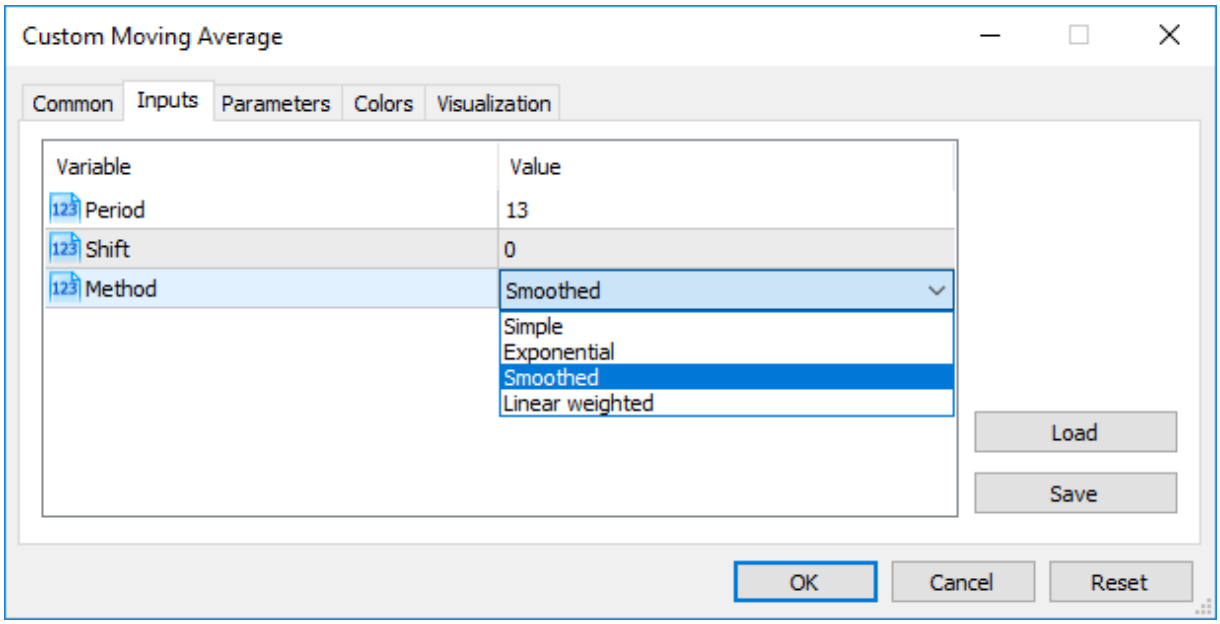
input bellek sınıfı dışsal değişkenleri tanımlar. Veri tipinden önce *input* şekillendiricisi belirtilir. Bu şekillendiriciye sahip olan değişkenler salt okunur durumdadırlar ve MQL5 programlarının içinde değiştirilemezler. Girdi değişkenlerinin değerleri sadece kullanıcı tarafından, program özellikleri penceresinde değiştirilebilir. OnInit() çağrısından hemen önce başlatılırlar.

Girdi değişkeni adlarının maksimum uzunluğu 63 karakterdir. string tipinde girdi parametresi için **maksimum** değer uzunluğu (dizge uzunluğu) 191 ila 253 karakter arasında olabilir (Nota bakın). Minimum uzunluk 0 karakterdir (değer ayarlanmamıştır).

Örnek:

```
//--- giriş parametreleri
input int             MA_Period=13;
input int             MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMMA;
```

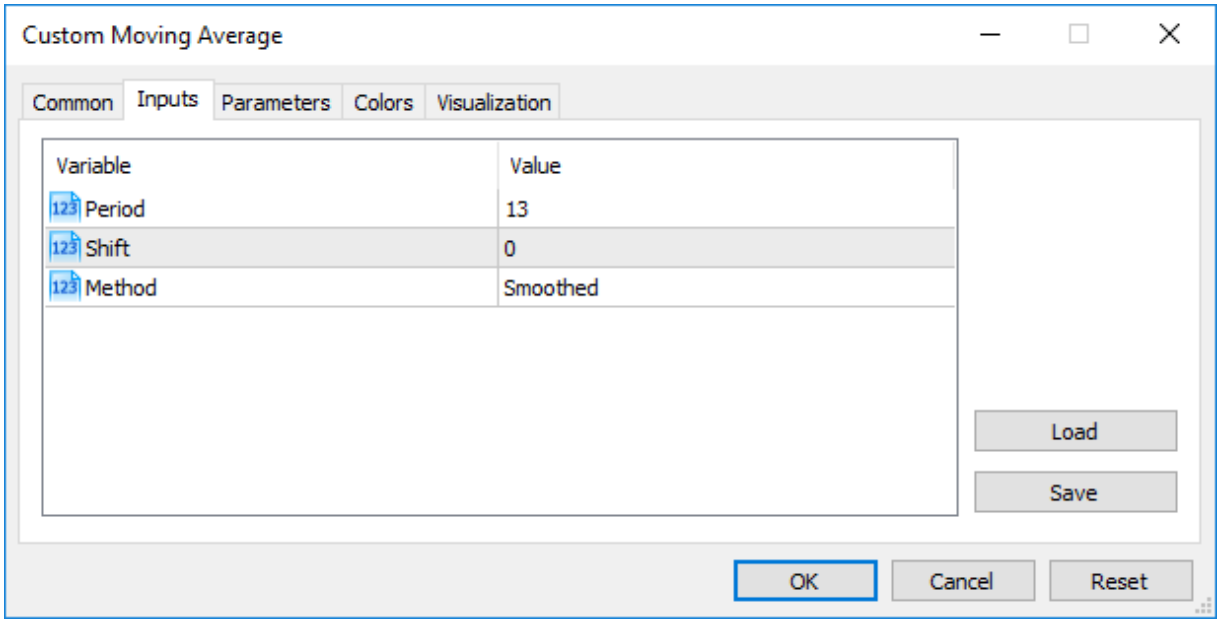
Girdi değişkenleri, programın giriş parametrelerini belirlerler. Programın özellikler penceresinde yer alırlar.



Parametre isimlerinin veriler sekmesindeki görünümünü farklı şekilde ayarlamak mümkündür. Bunu yapmak için yorumlar kullanılır. Metin, giriş parametresinin tarifinden sonra aynı satırda yer almalıdır. Bu şekilde giriş parametreleri için daha anlaşılır isimler oluşturulabilir.

Örnek:

```
//--- giriş parametreleri
input int             InpMAPeriod=13;         // Düzleştirme periyodu
input int             InpMAShift=0;           // Yatay çizgi kaydırma
input ENUM_MA_METHOD InpMAMethod=MODE_SMMA;  // Düzleştirme yöntemi
```



Not: Karmaşık tipli diziler ve değişkenler, giriş parametresi görevi üstlenemezler.

Not: Yorum dizgisinin boyutu 63 karakteri geçemez.

Not: string tipinde giriş değişkenleri için, değer uzunluğunun (dizge uzunluğu) sınırlaması aşağıdaki koşullar tarafından belirlenir:

- parametre değeri "parameter_name=parameter_value" dizgesi ile temsil edilir ('=' dikkate alınır),
- maksimum gösterim uzunluğu 255 karakterdir (*total_length_max*=255 veya '=' hariç 254 karakter),
- *parameter_name_length* dizge parametresinin maksimum uzunluğu 63 karakterdir.

Bu nedenle, bir dizge parametresi için maksimum dizge boyutu aşağıdaki denklem kullanılarak hesaplanır:

```
parameter_value_length=total_length_max-parameter_name_length=254-parameter_name_lengt
```

Bu, 191'den (parametre_name_length=63) 253 karaktere (parameter_name_length=1) kadar maksimum dizge boyutunu sağlar.

Parametrelerin, Özel Göstergeler MQL5 Programlarından Çağrıldığı Sırada Geçirilmesi

Özel göstergeler iCustom() fonksiyonu kullanılarak çağrılırlar. Özel göstergenin isminden sonra gelen parametreler, göstergenin girdi değişkenleriyle örtüşmelidir. Eğer belirtilen parametreler, çağrılan özel göstergedeki girdi değişkenlerden az ise, boş bırakılan parametreler, değişkenlerin bildirimi sırasında belirtilen değerler ile doldurulur.

Eğer özel gösterge, OnCalculate fonksiyonunun ilk tipini kullanıyorsa (gösterge aynı veri dizisi ile hesaplanıyorsa), göstergenin çağrılırken, son parametre olarak, ENUM_APPLIED_PRICE değerleri veya başka bir göstergenin tanıtıcı değeri kullanılmalıdır. Girdi değişkenlerine karşılık gelen tüm parametreler açıkça belirtilmelidir.

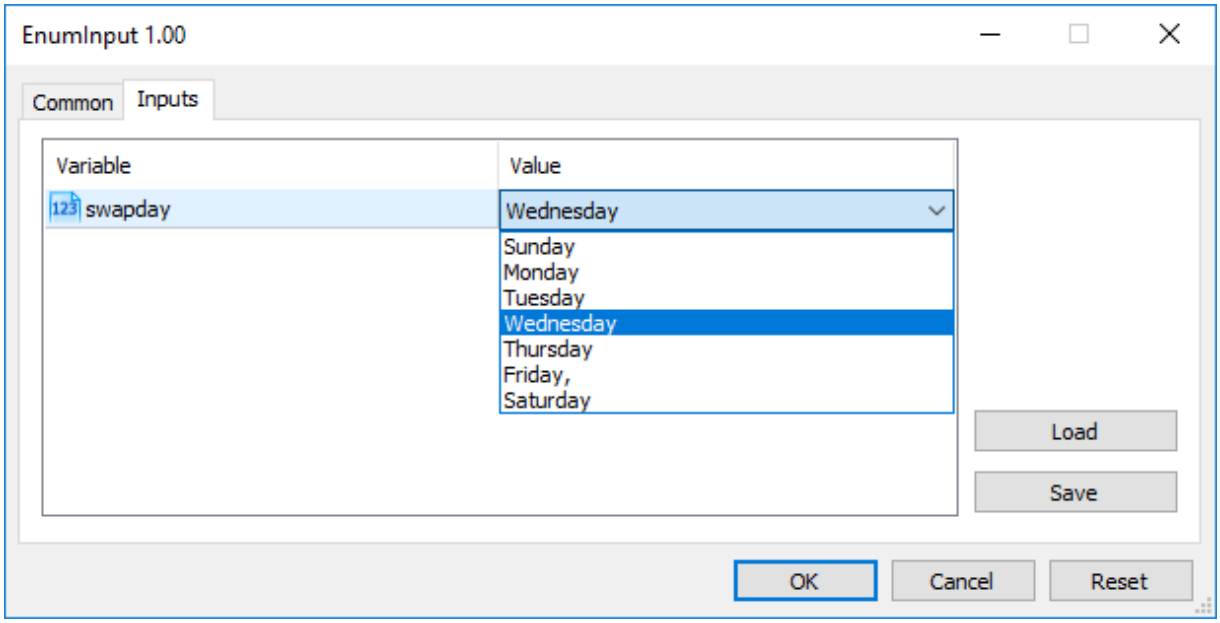
Giriş parametresi Şeklinde Sayımlar

MQL5 dilinde sadece yerleşik (gömülü) sayımlar değil, aynı zamanda kullanıcı tanımlı sayımlar da girdi değişkenleri (MQL5 programları için giriş parametreleri) olarak tanımlanabilir. Örneğin, haftanın günlerini tarif eden 'dayOfWeek' sayımını oluşturabiliriz ve bunu haftanın özel bir gününü belirleyecek bir girdi değişkeni şeklinde kullanabiliriz (bir sayı olarak değil, daha genel bir şekilde).

Örnek:

```
#property script_show_inputs
//--- haftanın günleri
enum dayOfWeek
  {
   S=0,      // Sunday
   M=1,      // Monday
   T=2,      // Tuesday
   W=3,      // Wednesday
   Th=4,     // Thursday
   Fr=5,     // Friday,
   St=6,     // Saturday
  };
//--- giriş parametreleri
input dayOfWeek swapday=W;
```

Betiğin başlatılması sırasında, kullanıcının gerekli değeri seçebilmesini sağlamak için, #property script_show_inputs önişlemci komutunu kullanırız. Betiği başlatırız ve listeden dayOfWeek sayımının değerlerinden birini seçebiliriz. 'EnumInInput' betiğini başlatırız ve veriler sekmesine gideriz. Varsayılan olarak, swap gününün değeri (üçlü swap günü) Çarşambadır (W = 3) ama başka bir değer belirleyebilir ve bu değeri program işlemini değiştirmek için kullanabiliriz.



Bir sayımın muhtemel değerlerinin sayısı sınırlıdır. Bir giriş değeri seçmek için çekme-liste kullanılır. listede gösterilen değerler için anımsatıcı isimler kullanılır. Örnekte görüldüğü gibi, herhangi bir yorum bu anımsatıcı isimle ilişkilendirilmişse, yorum içeriği anımsatıcı ismin yerine kullanılır.

'dayOfWeek' sayımının her bir değeri 0 ile altı arasında bir numaraya sahiptir ama parametre listesinde her bir değer için belirlenmiş yorumlar gözükür. Bu şekilde, programların giriş parametrelerinin açık tarifler ile yazılması için fazladan esneklik sağlanır.

sinput Şekillendiricili Değişkenler

input şekillendiricili değişkenler sadece program çalıştırıldığı zaman dışsal değişkenlerin ayarlanmasını sağlamaz, aynı zamanda Strateji Sınayıcıdaki alım-satım stratejilerinin optimizasyonu sırasında da gereklidirler. Her girdi değişkeni (string tipli olanlar hariç) optimizasyonda kullanılabilir.

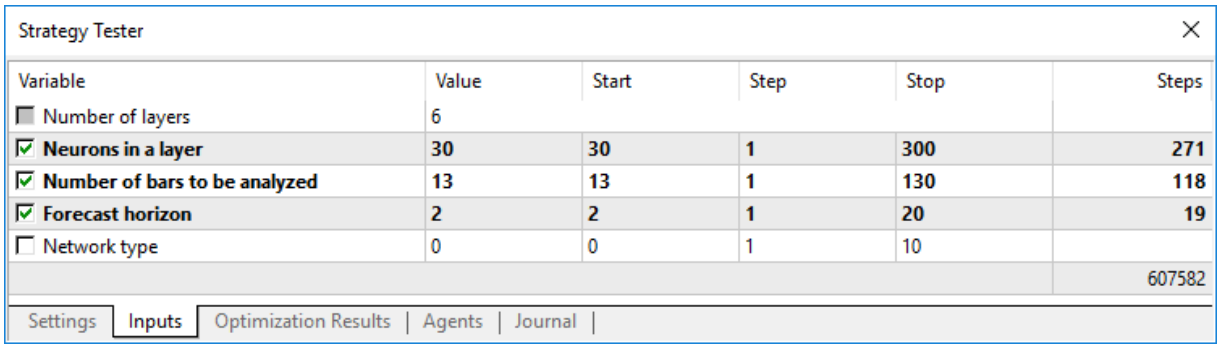
Bazen, bazı dışsal parametrelerin, sınayıcıdaki tüm geçişlerden çıkarılması gerekebilir. sinput bellek şekillendirici bu gibi durumlar için düşünülmüştür. sinput, statik dışsal değişken bildirimini simgeler (sinput = statik girdi). Yani, Uzman Danışman içindeki şu bildirimin

```
sinput       int layers=6;   // Katmanların sayısı
```

bütün bildirime eş değer olduğu anlamına gelir.

```
static input int layers=6;   // Katmanların sayısı
```

sinput şekillendiricisi ile bildirilen değişkenler MQL5 programının giriş parametreleridir. Bunların değerleri programın çalıştırılması sırasında değiştirilebilir. Ama ilgili değişkenler giriş parametrelerinin optimizasyonunda kullanılmaz. Başka bir deyişle, belli bir duruma uyan en iyi parametre kümesini ararken, bu değişkenin değerleri araştırılmaz.



Yukarıda gözüken Uzman Danışmanın 5 dışsal değişkeni bulunmaktadır. "Number of layers" parametresi sinput olarak bildirilmiştir ve 6'ya eşittir. Bu parametre, bir alım-satım stratejisi optimizasyonu sırasında değiştirilemez. Bunun için ileride kullanılacak gerekli değeri belirleyebiliriz. Başla, Adım ve Dur alanları bu tür değişkenler için mevcut değildir.

Bu yüzden, değişken için sinput şekillendiricisini belirlediğimizde, kullanıcılar bu parametreyi optimize edemeyecektir. Yani kullanıcılar, optimizasyon sırasında, belirlenen kapsamda otomatik numaralandırma için, strateji sınayıcı içinde başlangıç ve son değerleri belirleyemezler.

Ama bu kurala bir istisna bulunmaktadır: ParameterSetRange() fonksiyonu kullanılarak, sinput değişkenler üzerinde oynanabilir. Bu fonksiyon, statik girdi (sinput) şeklinde bildirilenler de dahil olmak üzere, herhangi bir giriş değişkeni için ayarlanan mevcut değerlerin program kontrolü için özellikle getirilmiştir. ParameterGetRange() fonksiyonu, optimizasyon başladığında, (OnTesterInit() işleyicisi içinde) girdi değişkenlerinin değerlerinin alınmasını sağlar. Ayrıca optimize edilmiş parametre değerlerinin numaralanacağı kapsamın ve değişim adımı değerinin sıfırlanmasına da izin verir.

Bu şekilde, giriş parametreleriyle çalışan iki fonksiyon ile sinput şekillendiricisini birleştirilir ve giriş parametrelerinin optimizasyon aralıklarının diğer giriş parametrelerinin değerlerine bağlı olarak ayarlandığı bir esneklik oluşturulması sağlanır.

Giriş parametrelerinin düzenlenmesi

MQL5 programlarıyla çalışırken kolaylık için, giriş parametreleri group anahtar sözcüğü kullanılarak adlandırılmış bloklara bölünebilir. Bu, bazı parametrelere gömülü mantığa dayanarak diğerlerinden görsel olarak ayrılmasını sağlar.

```
input group           "Grup adı"
input int             variable1 = ...
input double          variable2 = ...
input double          variable3= ...
```

Böyle bir bildirimden sonra; tüm girdi parametreleri, bir uygulama grafikte veya strateji sınayıcısında başlatılırken parametre konfigürasyonunu MQL5 kullanıcıları için basitleştiren belirli bir gruba görsel olarak birleştirilir. Her bir grup yeni bir grup bildirimi gelene kadar geçerlidir:

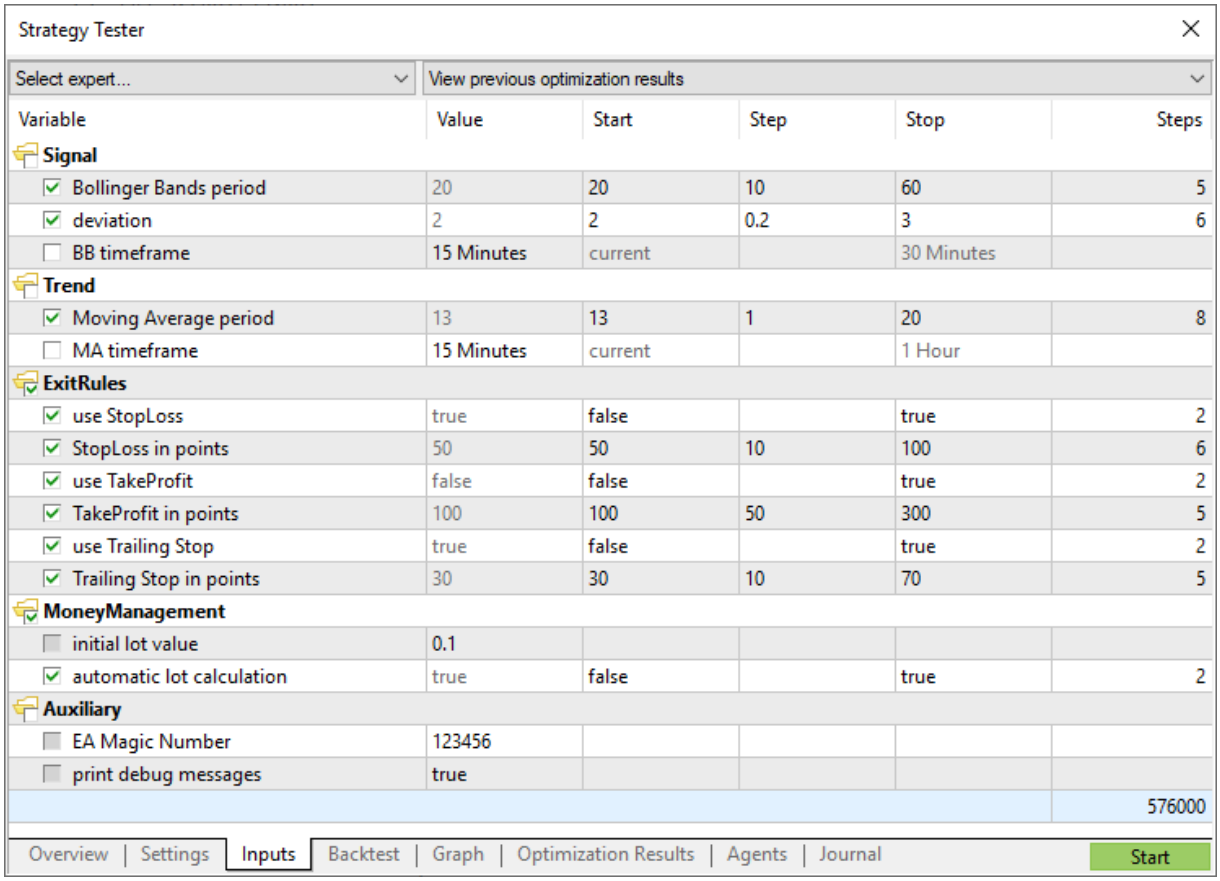
```
input group           "Grup adı #1"
input int             group1_var1 = ...
input double          group1_var2 = ...
input double          group1_var3 = ...

input group           "Grup adı #2
input int             group2_var1 = ...
input double          group2_var2 = ...
input double          group2_var3 = ...
```

Giriş parametreleri bloğunun amaçlarına göre bölündüğü örnek bir Uzman Danışman:

```
input group           "Signal"
input int             ExtBBPeriod    = 20;       // Bollinger Bantları periyodu
input double          ExtBBDeviation= 2.0;       // sapma
input ENUM_TIMEFRAMES ExtSignalTF=PERIOD_M15;    // BB zaman aralığı

input group           "Trend"
input int             ExtMAPeriod    = 13;       // Hareketli Ortalama (MA) periyodu
input ENUM_TIMEFRAMES ExtTrendTF=PERIOD_M15;     // MA zaman aralığı

input group           "ExitRules"
input bool            ExtUseSL       = true;     // Zararı Durdur (SL) kullan
input int             Ext_SL_Points = 50;        // puan cinsinden SL
input bool            ExtUseTP       = false;    // Kar Al (TP) kullan
input int             Ext_TP_Points = 100;       // puan cinsinden TP
input bool            ExtUseTS       = true;     // Takip Eden Zarar Durdurucu kullan
input int             Ext_TS_Points = 30;        // puan cinsinden Takip Eden Zarar Du

input group           "MoneyManagement"
```

```
sinput double          ExtInitialLot = 0.1;       // başlangıç lot değeri
input bool             ExtUseAutoLot = true;      // otomatik lot hesaplama

input group            "Auxiliary"
sinput int             ExtMagicNumber = 123456;   // Uzman Danışman Sihirli Sayısı
sinput bool            ExtDebugMessage= true;     // hata ayıklama iletilerini yazdır
```

Strateji sınayıcısında bu gibi bir Uzman Danışmanı başlatırken, giriş parametreleri bloğunu daraltmak/genişletmek için bir grup adına çift tıklayabilir, ayrıca optimizasyon için grubun tüm parametrelerini seçmek üzere grup onay kutusuna tıklayabilirsiniz.



Variable	Value	Start	Step	Stop	Steps
Signal					
Bollinger Bands period	20	20	10	60	5
deviation	2	2	0.2	3	6
BB timeframe	15 Minutes	current		30 Minutes	
Trend					
Moving Average period	13	13	1	20	8
MA timeframe	15 Minutes	current		1 Hour	
ExitRules					
use StopLoss	true	false		true	2
StopLoss in points	50	50	10	100	6
use TakeProfit	false	false		true	2
TakeProfit in points	100	100	50	300	5
use Trailing Stop	true	false		true	2
Trailing Stop in points	30	30	10	70	5
MoneyManagement					
initial lot value	0.1				
automatic lot calculation	true	false		true	2
Auxiliary					
EA Magic Number	123456				
print debug messages	true				
					576000

Ayrıca Bakınız

iCustom, Sayımlar, Programların özellikleri

Extern Değişkenler

Anahtar sözcük extern, statik bellek sınıfının tanıtıcıları gibi değişken tanıtıcılarını global kapsamla bildirmek için kullanılır. Bu değişkenler programın başından itibaren vardır ve bunlar için gereken bellek, programın başladıktan hemen sonra tahsis edilir.

Birden çok kaynak dosya içeren programlar oluşturabilirsiniz; bu durumda #include önişlemcisine yapılan bir direktif kullanılır. Aynı tipe ve tanımlayıcıya sahip olan ve 'extern' şeklinde bildirilen değişkenler, aynı proje içinde farklı kaynak dosyalarında bulunabilir.

Projenin tamamı derlenirken, aynı tipe ve bir tanımlayıcıya sahip tüm 'extern' değişkenler, global değişken havuzunun hafızasının bir bölümü ile ilişkilendirilir. Extern değişkenler, kaynak dosyalarının ayrı olarak derlenmesi için kullanışlıdır. Bir defaya mahsus olmak üzere başlatılabilirler - birkaç kez başlatılan, aynı tipe ve aynı tanımlayıcıya sahip extern değişkenlerin varlığına izin verilmez.

Ayrıca Bakınız

Veri Tipleri, Tiplerin Kapsüllenmesi ve Genişletilebilirliği, Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Oluşturulması ve Silinmesi

Değişkenlerin Başlatılması

Her değişken, tanımlama sırasında başlatılabilir. Değişken açık yolla başlatılmamışsa, değişkende saklanan değer her şey olabilir. Gizli başlatma kullanılmaz.

Global ve statik değişkenler sadece karşılık gelen tipte bir sabitle veya bir sabit ifade ile başlatılabilirler. Yerel değişkenler sadece sabitlerle değil, herhangi bir ifadeyle de başlatılabilirler.

Global ve statik değişkenlerin başlatılma işlemi sadece bir kez gerçekleştirilir. Yerel değişkenlerin başlatılması ise, karşılık gelen fonksiyonu her çağırışınızda gerçekleştirilir

Örnekler:

```
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 } };
//--- tetris'den
int    right[4]={WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER,
                 WIDTH_IN_PIXELS+VERT_BORDER,WIDTH_IN_PIXELS+VERT_BORDER};
//--- yapının tüm alanlarının sıfır değerleriyle başlatılması
MqlTradeRequest request={};
```

Dizi elemanlarının değerlerinin listesi, küme parantezlerinin içine iliştirilmiş olmalıdır. Unutulmuş başlatma dizilimleri sıfıra eşit kabul edilir.

Başlatılan dizinin büyüklüğü belirtilmemişse, başlatma diziliminin büyüklüğüne göre derleyici tarafından belirlenir.

Örnekler:

```
struct str3
  {
   int               low_part;
   int               high_part;
  };
struct str10
  {
   str3              s3;
   double            d1[10];
   int               i3;
  };
void OnStart()
  {
   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]);
  }
```

Yapı tipli değişkenlerin de statik dizilerde olduğu gibi -kapalı yolla ayarlanmış bir büyüklükle- kısmi olarak başlatılmalarına izin verilir. Bir yapı veya dizinin ilk elemanlarını bir veya daha çok defa başlatabilirsiniz, bu durumda diğer elemanlar sıfır ile başlatılacaktır.

Ayrıca Bakınız

Veri Tipleri, Tiplerin Kapsüllenmesi ve Genişletilebilirliği, Değişkenlerin Görünürlük Alanları ve Ömürleri, Nesnelerin Oluşturulması ve Silinmesi

Değişkenlerin Görünürlük Alanları ve Ömürleri

İki temel kapsam türü vardır: yerel kapsam ve global kapsam.

Fonksiyonların dışında bildirilen değişkenler global kapsamda yer alır. Bu gibi değişkenlere programın her yerinden erişilebilir. Bunlar belleğin global havuzuna yerleştirilirler, bu yüzden ömürleri de programın ömrüyle aynıdır.

Bir blok (programın küme parantezleri içine yerleştirilen parçası) içinde bildirilen değişkenler yerel kapsama sahiptir. Bu tür değişkenler bildirildiği blok dışında görünmezdir (dolayısıyla mevcut değildir). Yerel bildirimin en bilindik örneği fonksiyon içinde bildirilen değişkenlerdir. Yerel olarak bildirilen bir değişken, yığın üzerinde yer alır ve bu değişkenin ömrü fonksiyonun ömrüne eşittir.

Yerel değişkenin kapsamı bildirildiği blokla sınırlı olduğundan, farklı bloklarda aynı isimde değişkenler bildirmek mümkündür;. Bu, global düzeye kadar daha üst seviyelerde bildirilmiş olan değişkenler için de geçerlidir.

Örnek:

```
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;
      //--- ilk sınır çubukları için boş değer ayarla
      for(i=0; i<limit; i++) LineBuffer[i]=0.0;
      //--- görünen ilk değeri hesapla
      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;
     }
//---
  }
```

Aşağıdaki satırda bildirilen i değişkenine dikkat edin

```
      for(int i=begin; i<limit; i++)
        {
         int k=i-begin+1;
         weightsum+=k;
         firstValue+=k*price[i];
        }
```

Bu değişkenin kapsamı sadece döngüyle sınırlıdır. Döngü dışında aynı isimle fonksiyon içinde bildirilmiş bir değişken bulunmaktadır. Ayrıca döngü içinde bildirilen k değişkeninin de kapsamı yine döngü gövdesidir.

Yerel değişkenler static erişim belirteci ile bildirilebilir. Bu durumda derleyici global bellek havuzunda bir değişkene sahip olur. Bu yüzden bir statik değişkenin ömrü programın ömrüne eşittir. Burada bu değişkenin kapsamı bildirildiği blok ile sınırlıdır.

Ayrıca Bakınız

Veri Tipleri, Tiplerin Kapsüllenmesi ve Genişletilebilirliği, Değişkenlerin Başlatılması, Nesnelerin Oluşturulması ve Silinmesi

Nesnelerin Oluşturulması ve Silinmesi

Bir MQL5 programı çalıştırılmak için yüklendiğinde, her değişkene tipine göre bellek tahsis edilir. Erişim düzeylerine göre değişkenler global değişkenler ve yerel değişkenler olmak üzere iki türe ayrılır. Bellek sınıfına göre bunlar, bir MQL5 programının statik ve otomatik giriş parametreleri olabilirler. Her bir değişken, eğer gerekliyse karşılık gelen bir değer ile başlatılır. Değişken, kullanımının ardından sonlandırılır ve onun tarafından kullanılan bellek MQL5 yönetici sistemine verilir.

Global Değişkenlerin Başlatılması ve Sonlandırılması

Global değişkenler, bir MQL5 programının yüklenmesinin hemen ardından ve herhangi bir fonksiyon çağrılmadan önce otomatik olarak başlatılırlar. Başlatma sırasında ilk değerler basit tiplere atanır ve nesneler için -eğer varsa- bir yapıcı çağrılır. Girdi değişkenleri her zaman global düzeyde bildirilir ve program başlangıcında kullanıcı tarafından iletişim kutusunda ayarlanan değerler ile başlatılır.

Statik değişkenler genellikle yerel seviyede başlatılmalarına rağmen, bunlar için bellek önceden tahsis edilir. Global değişkenlerde olduğu gibi, başlatma işlemi programın yüklenmesinin hemen ardından gerçekleştirilir.

Başlatma sırası değişkenlerin bildirim sırasına karşılık gelir. Sonlandırma işlemi ise ters sırada gerçekleştirilir. Bu kural sadece 'new' operatörü ile oluşturulmamış değişkenler için doğrudur. Böyle değişkenler programın yüklenmesiyle oluşturulup başlatılır ve programın kaldırılmasıyla sonlandırılır.

Yerel Değişkenlerin Başlatılması ve Sonlandırılması

Yerel düzeyde başlatılan bir değişken statik değilse, değişken için otomatik olarak bellek tahsis edilir. Yerel değişkenler de tıpkı global değişkenler gibi, program yönetimi yerel değişkenin bildirimiyle karşılaştığı anda otomatik olarak başlatılırlar. Yani, başlatma sırası bildirim sırasına karşılık gelir.

Yerel değişkenler bildirildikleri program bloğunun bitiminde, bildirim sıralarının tersine göre sonlandırılır. Program bloğu bir birleşik operatördür; switch seçim operatörünün, döngü operatörlerinin (for, while, do-while), bir fonksiyon gövdesinin veya bir if-else operatörünün parçası olabilir.

Yerel değişkenler sadece program yönetimi yerel değişken bildirimiyle karşılaştığı anda başlatılırlar. Eğer programın çalıştırılması sırasında değişkenin bildirildiği blok çalıştırılmazsa, değişken de çalıştırılmaz.

Dinamik Olarak Yerleştirilmiş Nesnelerin Başlatılması ve Sonlandırılması

Nesne işaretçileri için özel bir durum söz konusudur; işaretçinin bildirimi, karşılık gelen nesnenin başlatılmasını gerektirmez. Dinamik olarak yerleştirilmiş nesneler sadece, sınıf örneği new operatörü ile oluşturuldığı zaman başlatılırlar. Nesnelerin başlatılması, karşılık gelen sınıfın bir yapıcısının çağrılmasını gerektirir. Eğer sınıf içinde karşılık gelen bir yapıcı yoksa, bu sınıfın basit tipli üyeleri otomatik olarak başlatılmayacaktır. Sınıfın dizgileri, dinamik dizileri ve nesneleri ise otomatik olarak başlatılır.

İşaretçiler global veya yerel düzeyde bildirilebilirler. Ayrıca boş değer NULL ile veya, aynı tipin yada miras tipin işaretçi değeri ile başlatılabilirler. Eğer *new* operatörü, yerel düzeyde bildirilmiş bir işaretçi için çağrılırsa, kapsamdan çıkmadan önce işaretçinin silinmesi için *delete* operatörü kullanılmalıdır. Aksi durumda işaretçi kaybolur ve nesnenin açık yolla silinmesi başarısız olur.

object_pointer=new *Class_name*, ifadesi ile oluşturulmuş tüm nesneler, sonradan delete(*object_pointer*) operatörü kullanılarak silinmelidir. Herhangi bir sebepten ötürü, böyle bir değişken delete operatörü ile silinmezse; program tamamlandığında buna karşılık gelen kayıt girdisi, "Uzmanlar" sekmesinde gözükecektir. Birkaç değişkeni bildirip, ardından hepsine tek bir sınıfın işaretçisini atamak mümkündür.

Eğer dinamik olarak oluşturulmuş bir nesne bir yapıcıya sahipse, bu yapıcı *new* operatörünün çalıştırılmasıyla beraber çağrılacaktır. Eğer nesne bir yıkıcıya sahipse, yapıcı, *delete* operatörünün çalıştırılmasıyla çağrılacaktır.

Bu şekilde dinamik olarak yerleştirilmiş nesneler, sadece *new* operatörü ile oluşturulacak ve *delete* operatörü ile (veya programın kaldırılması sırasında MQL5 yönetici sistemi tarafından otomatik olarak) silinecektir. Dinamik olarak oluşturulan nesnelerin bildirim sıraları, başlatılma sıralarını etkilemez. Başlatma ve sonlandırma sırası, tamamen programcı tarafından kontrol edilir.

MQL5 dilinde dinamik bellek tahsisi

Dinamik dizilerle çalışılırken serbest bırakılan bellek alanı hemen işletim sistemine devredilir.

new operatörü kullanarak dinamik sınıf nesneleri oluştururken, gereken bellek ilk olarak bellek yöneticisinin çalıştığı sınıf bellek havuzundan talep edilir. Eğer havuzda yeterli bellek mevcut değilse, gereken bellek alanı işletim sisteminden istenir. Bir dinamik nesnenin delete operatörü ile silinmesinin hemen ardından, serbest kalan bellek sınıf bellek havuzuna devredilir.

Bellek yöneticisi; OnInit(), OnDeinit(), OnStart(), OnTick(), OnCalculate(), OnTimer(), OnTrade(), OnTester(), OnTesterInit(), OnTesterPass(), OnTesterDeinit(), OnChartEvent(), OnBookEvent() olay işleme fonksiyonlarının sonlandırılmasının ardından, serbest kalan belleği işletim sistemine devreder.

Kısaca Değişken Nitelikleri

Oluşturulma ve silinme sırası, yapıcı ve yıkıcıların çağrılması gibi konularla ilgili temel bilgiler aşağıdaki tabloda verilmiştir.

	Global otomatik değişken	Yerel otomatik değişken	Dinamik olarak oluşturulmuş nesne
Başlatma	MQL5 programının yüklenmesinin hemen ardından	çalıştırma esnasında, bildirildiği satıra ulaşıldığında	**new** operatörünün çalıştırılmasında
Başlatma sırası	bildirim sırasına göre	bildirim sırasına göre	bildirim sırasından bağımsız
Sonlandırma	MQL5 programı kaldırılmadan önce	bildirim bloğundan çıkıldığında	**delete** operatörü çalıştırıldığında veya MQL5 programı kaldırıldığında
Sonlandırma sırası	başlatma sırasının tersine	başlatma sırasının tersine	başlatma sırasından bağımsız

	Global otomatik değişken	Yerel otomatik değişken	Dinamik olarak oluşturulmuş nesne
Yapıcı çağrısı	MQL5 programı yüklendiğinde	başlatılma sırasında	*new* operatörünün çalıştırılmasında
Yıkıcı çağrısı	MQL5 programının kaldırılması sırasında	değişkenin başlatıldığı bloktan çıkıldığında	*delete* operatörünün çalıştırılmasıyla
Hata günlükleri	"Uzmanlar" günlüğünde, otomatik olarak oluşturulmuş bir nesnenin silinme denemesinin başlatıldığına dair günlük mesajı	"Uzmanlar" günlüğünde, otomatik olarak oluşturulmuş bir nesnenin silinme denemesinin başlatıldığına dair günlük mesajı	"Uzmanlar" günlüğünde, MQL5 programı kaldırılırken silinmemiş olan dinamik oluşturulmuş nesnelere dair günlük mesajı

Ayrıca Bakınız

Veri Tipleri, Tiplerin Kapsüllenmesi ve Genişletilebilirliği, Değişkenlerin Başlatılması, Değişkenlerin Görünürlük Alanları ve Ömürleri

Önişlemci

Önişlemci, MQL5 derleyicisinin özel bir alt sistemidir. Kaynak kodunun program derlenmeden önce hızlı bir şekilde hazırlanması için tasarlanmıştır.

Kaynak kodunun okunabilirliğini artırır. Uygulamalar, MQL5 programlarının kaynak kodlarını taşıyan belirli dosyaları içerecek şekilde yapılandırılabilir. Belirli sabitlere anımsatıcı isimler verilmesi okunabilirliğin artırılmasına yardımcı olur.

Önişlemci MQL5 programlarının özel parametrelerinin belirlenmesini de sağlar:

- Sabitleri belirler
- Program özelliklerini ayarla
- Program metnine dosya ekle
- Dışarıdan fonksiyon al
- Conditional Compilation

Önişlemci yönergeleri derleyici tarafından derlenmeden önce kaynak kodunu ön işleme koymak için kullanılır. Yönerge her zaman # ile başlar, bu nedenle derleyici değişkenlerin, fonksiyonların vb. simlerinde sembol kullanımını yasaklar.

Her direktif ayrı bir girdi ile tanımlanır ve satır sonuna kadar geçerlidir. Bir girişte birkaç yönergeyi kullanamazsınız. Yönerge girişi çok büyükse, '\' simgesini kullanarak birkaç satıra bölünebilir. Bu durumda, bir sonraki satır direktif girişinin bir devamı olarak kabul edilir.

```
//+------------------------------------------------------------------+
//|  foreach sözde-operatorü                                         |
//+------------------------------------------------------------------+
#define ForEach(index, array) for (int index = 0,                    \
   max_##index=ArraySize((array));                                   \
   index<max_##index; index++)
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   string array[]={"12","23","34","45"};
//--- ForEach kullanarak diziyi atlamak
   ForEach(i,array)
     {
      PrintFormat("%d: array[%d]=%s",i,i,array[i]);
     }
  }
//+------------------------------------------------------------------+
/* Output result
   0: array[0]=12
   1: array[1]=23
   2: array[2]=34
   3: array[3]=45
*/
```

Derleyici için, bu üç #define yönergesi satırı tek bir uzun satır gibi görünür. Yukarıdaki örnek, #define makrolarında kullanılan bir birleştirme operatörü olan ## karakteri de iki makro belirteçlerini birleştirmek için de geçerlidir. İşaretçiler birleştirme işleci, bir makro tanımında ilk veya son olamaz.

Sabitlerin Bildirimi (#define, #undef)

Önişlemci yönergeleri derleyici tarafından derlenmeden önce kaynak kodunu ön işleme koymak için kullanılır. Yönerge her zaman # ile başlar, bu nedenle derleyici değişkenlerin, fonksiyonların vb. simlerinde sembol kullanımını yasaklar.

Her direktif ayrı bir girdi ile tanımlanır ve satır sonuna kadar geçerlidir. Bir girişte birkaç yönergeyi kullanamazsınız. Yönerge girişi çok büyükse, '\' simgesini kullanarak birkaç satıra bölünebilir. Bu durumda, bir sonraki satır direktif girişinin bir devamı olarak kabul edilir.

#define direktifi sabitlere anımsatıcı isimler verilmesi için kullanılır. İki biçimi vardır:

```
#define tanımlayıcı ifade                       // parametresiz biçim
#define tanımlayıcı(par1,... par8) ifade        // parametrik biçim
```

#define direktifi, **ifadeyi** kaynak metinde bulunan tüm *tanımlayıcı* girişleri için ikame eder. *Tanımlayıcı*, sadece ayrı bir simge olarak kullanılıyorsa ikame edilir. Bir yorumun, bir dizginin veya daha uzun başka bir tanımlayıcının parçasıysa ikame edilmez.

Sabit tanımlayıcı, değişken isimleri ile aynı kurallara göre yönetilir. Değer her tipten olabilir:

```
#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");
  }
```

ifade; anahtar sözcükler, sabitler, sabit veya sabit olmayan ifadeler gibi çeşitli simgeden oluşabilir. Satır sonu ile biter ve yeni satıra aktarılamaz.

Örnek:

```
#define TWO         2
#define THREE       3
#define INCOMPLETE TWO+THREE
#define COMPLETE   (TWO+THREE)
void OnStart()
  {
   Print("2 + 3*2 = ",INCOMPLETE*2);
   Print("(2 + 3)*2 = ",COMPLETE*2);
  }
// Sonuç
// 2 + 3*2 = 8
// (2 + 3)*2 = 10
```

Parametrik Biçim #define

Parametrik biçimde, kaynak metinde bulunan tüm tanımlayıcı girişleri, daha sonra gerçek parametreler hesaba katılarak, ifadeyle değiştirilir. Örneğin:

```
 // iki parametreli (a ve b) örnek
#define A 2+3
#define B 5-1
#define MUL(a, b) ((a)*(b))

double c=MUL(A,B);
Print("c=",c);
/*
ifade double c=MUL(A,B);
eşdeğerdir double c=((2+3)*(5-1));
*/
// Sonuç
// c=20
```

Parametreli ifadeler kullanırken parametreleri parantez içine yazdığınızdan emin olun. Bu yeterince açık olmayan, bulunması zor hataların önlenmesine yardımcı olacaktır. Kodu parantezler olmadan yeniden yazarsak sonuç farklı olur:

```
 // iki parametreli (a ve b) örnek
#define A 2+3
#define B 5-1
#define MUL(a, b) a*b

double c=MUL(A,B);
Print("c=",c);
/*
ifade double c=MUL(A,B);
eşdeğer double c=2+3*5-1;
*/
// Sonuç
// c=16
```

Parametrik biçim kullanılırken, en fazla 8 karaktere izin verilir.

```
 // doğru parametrik biçim
#define LOG(text)  Print(__FILE__,"(",__LINE__,") :",text)   // tek parametre - 'text

 // yanlış parametrik biçim form
#define WRONG_DEF(p1, p2, p3, p4, p5, p6, p7, p8, p9)   p1+p2+p3+p4 // p1'den p9'a kad
```

#undef direktifi

#undef direktifi daha önce tanımlanan makro ikamesinin bildirimini iptal eder.

Örnek:

```
#define MACRO
```

```
void func1()
  {
#ifdef MACRO
   Print("MACRO is defined in ",__FUNCTION__);
#else
   Print("MAKRO şu fonksiyonunda tanımlanmış: ",__FUNCTION__);
#endif
  }

#undef MACRO

void func2()
  {
#ifdef MACRO
   Print("MAKRO şu fonksiyonunda tanımlanmış: ",__FUNCTION__);
#else
   Print("MAKRO şu fonksiyonunda tanımlanmış: ",__FUNCTION__);
#endif
  }

void OnStart()
  {
   func1();
   func2();
  }

/* Sonuç:
MAKRO şu fonksiyonda tanımlanmış: func1
MAKRO şu fonksiyonda tanımlanmamış func2
*/
```

Ayrıca Bakınız

Tanımlayıcılar, Karakter Sabitleri

Program Özellikleri (#property)

Her MQL5 programı #property olarak isimlendirilen özel ek parametreler belirlemeye olanak sağlar. Bu parametreler, onları açıkça çalıştırma gereksinimi duymayan programlar için terminale yardım ederler. Bu, her şeyden önce göstergelerin dışsal ayarları ile ilgilidir. Dahil edilen dosyalarda tarif edilmiş olan özellikler gözardı edilir. Özellikler ana MQL5 dosyasında belirtilmiş olmalıdır.

```
#property tanımlayıcı değer
```

Derleyici bildirilen değerleri çalıştırılan modül düzenlenirken yazacaktır.

Sabit	Tip	Açıklama
ikon	string	EX5 programının ikonu olarak kullanılacak görüntü dosyasının adresi. Dosya adresi belirleme kuralları kaynaklarda olduğu gibidir. Özellik, MQL5 kaynak kodunun ana modülünde belirtilmelidir. İkon dosyası ICOformatında olmalıdır.
link	string	Firmanın internet sayfasının linki
copyright	string	Firma adı
versiyon	string	Program versiyonu (en fazla 31 karakter)
açıklama	string	Bir MQL5 programının kısa açıklaması. Birkaç *açıklama* yazılabilir. Her biri, metnin bir satırını oluşturur. *Açıklamanın* toplam uzunluğu -satır atlama operatörü de dahil olmak üzere- 511 karakteri geçemez.
stacksize	int	MQL5 program yığınının boyutu. Tekrar eden fonksiyon çağrıları kullanırken uygun boyutta bir yığın gerekir. Çizelge üzerinde bir Uzman veya bir betik dosyası çalıştırılırken, en az 8 MB 'lık yığın alanı tahsis edilir. Göstergeler için kullanılan yığının boyutu 1 MB ile sabittir. Strateji sınayıcı içinde çalıştırılan programlar için her zaman 16 MB yığın alanı tahsis edilir.
kütüphane		Kütüphane anlamına gelir. Start (başlat) fonksiyonu bir giriş noktası (olay işleyicisi) yoktur. dışa aktarım şekillendiricisi ile fonksiyonların diğer MQL5 programlarından içe aktarımı yapılabilir
indicator_applied_price	int	"Uygula" alanı (gösterge özellikleri penceresi) için varsayılan değeri belirtir. ENUM_APPLIED_PRICE değerlerinden birini kullanabilirsiniz. Eğer özellik belirtilmemişse varsayılan değer PRICE_CLOSE olur
indicator_chart_window		Göstergeyi çizelge penceresinde göster
indicator_separate_window		Göstergeyi ayrı bir pencerede göster

Sabit	Tip	Açıklama
indicator_height	int	Gösterge alt penceresinin sabit büyüklüğü - piksel olarak (INDICATOR_HEIGHT özelliği)
indicator_buffers	int	Gösterge hesabı için kullanılacak tamponların sayısı
indicator_plots	int	Göstergedeki grafik serilerinin
indicator_minimum	double	Ayrı gösterge penceresi için ölçekleme taban limiti
indicator_maximum	double	Ayrı gösterge penceresi için ölçekleme tavan limiti
indicator_labelN	string	Veri Penceresinde görünen N-inci grafik serisi için etiket ayarlar. Birden çok gösterge tamponu gerektiren grafik serileri için (DRAW_CANDLES, DRAW_FILLING ve diğerleri) etiket isimleri ';' ayracı ile tanımlanır.
indicator_colorN	color	N çizgisinin görüntüleneceği renk. N burada grafik serisinin numarasıdır. Numaralama 1'den başlar
indicator_widthN	int	Grafik serisi için çizgi kalınlığı. N burada grafik serisinin numarasıdır. Numaralama 1'den başlar
indicator_styleN	int	Grafik serisi için ENUM_LINE_STYLE değerleriyle belirlenen çizim stili. N - grafik serisinin numarası, numaralama 1'den başlar
indicator_typeN	int	Grafiksel çizim tipi, ENUM_DRAW_TYPE sayımının değerleriyle belirlenir. N - grafik serisinin numarası, numaralama 1'den başlar
indicator_levelN	double	N'nin, ayrı gösterge penceresindeki yatay seviyesi
indicator_levelcolor	color	Göstergenin yatay seviyelerinin rengi
indicator_levelwidth	int	Göstergenin yatay seviyelerinin kalınlığı
indicator_levelstyle	int	Göstergenin yatay seviyelerinin çizim stili
script_show_confirm		Betiği çalıştırmadan önce, bir doğrulama penceresi göster
script_show_inputs		Betiği çalıştırmadan önce, özelliklerin olduğu bir pencere göster ve bu doğrulama penceresini işlevsiz hale getir
tester_indicator	string	"*indicator_name.ex5*" şeklinde verilen gösterge ismi. Eğer karşılık gelen parametre bir dizgi sabiti ile ayarlanmışsa, sınama gerektiren göstergeler, iCustom() fonksiyonunun çağrısından otomatik olarak tanımlanır. Diğer tüm durumlarda (IndicatorCreate() fonksiyonunun kullanımı veya sabit olmayan bir dizginin, fonksiyon ismini belirleyen parametrede kullanımı durumlarında) bu özellik gereklidir

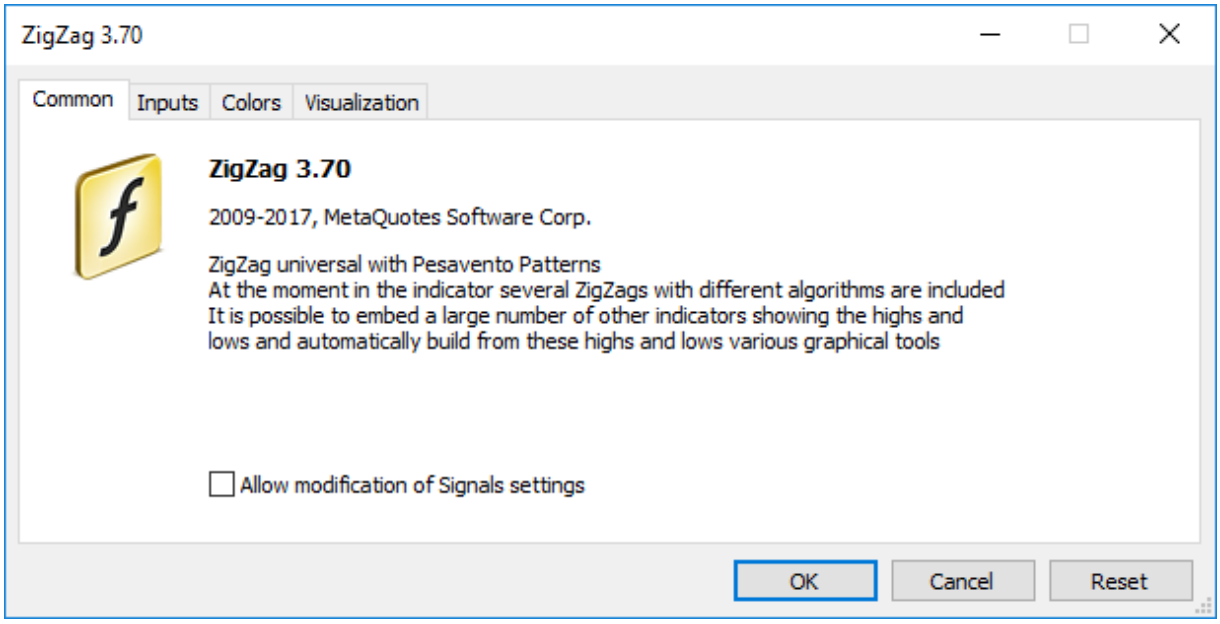
Sabit	Tip	Açıklama
tester_file	string	Sınama aracı için, uzantısı belirtilmiş şekilde, çift tırnak içinde (bir sabit dizgi olarak) dosya adı. Belirtilen dosya sınayıcıya aktarılır. Test edilecek girdi dosyaları - gerekli olanlar her zaman belirtilmelidir.
tester_library	string	Çift tırnak içinde, uzantılı olarak kütüphane adı. Bir kütüphane dll veya ex5 uzantılarına sahip olabilir. sınama gerektiren kütüphaneler otomatik olarak tanımlanır. Ama, kütüphanelerden herhangi biri özel göstergeler içinde kullanılırsa, bu özelik gereklidir
tester_set	string	Set dosyasının adı ile input parametresinin değeri ve adımı. Dosya test ve optimizasyondan önce tester'a gönderilir. Dosya adı bir uzantı ve bir çift teklif ile sabit string olarak belirtilmiştir. Eğer EA ismini ve versiyon numarasını bir set dosyası içinde "<expert_name>_<number>.set" olarak ayarlarsanız, ardından <number> sürüm numarası altında otomatik olarak parametre sürümleri download menüsüne eklenir. Örneğin, "MACD Sample_4.set" adı, "MACD Sample.mq5" EA ile versiyon numarası 4'e eşit olan bir set dosyası anlamına gelir. Format çalışmak için, test/optimizasyon ayarlarını elle strateji tester'a kaydetmenizi ve ardından bu şekilde oluşturulan set dosyasını açmanızı öneririz.
tester_no_cache	string	Optimizasyon uygulanırken, strateji sınayıcısı gerçekleşmiş olan tüm geçişlerin sonuçlarını optimizasyon önbelleğine kaydeder ve test sonucu da her bir girdi parametreleri kümesi için kaydedilir. Bu durum, yeniden hesaplama ile vakit kaybetmeden, hazır olan sonuçların aynı parametreler üzerindeki yeniden optimizasyon sırasında kullanılmasına olanak tanır. Ancak bazı görevlerde (örneğin, matematik hesaplamalarında), hesaplamaların optimizasyon önbelleğindeki hazır sonuçların kullanılabilirliğinden bağımsız olarak yapılması gerekebilir. Bu durumda, dosya *tester_no_cache* özelliğini içermelidir. Test sonuçları yine de önbellekte saklanır, böylece strateji sınayıcısında gerçekleştirilen geçişler hakkındaki tüm verileri görebilirsiniz.
tester_everytick_calculate	string	Strateji Sınayıcısında göstergeler sadece verilerine erişildiğinde, yani gösterge tamponlarının değerleri

Sabit	Tip	Açıklama
		talep edildiğinde hesaplanır. Her tik üzerinde gösterge değerleri elde etmeniz gerekmiyorsa, bu, çok daha hızlı bir test ve optimizasyon hızı sağlar. *tester_everytick_calculate* özelliğini belirterek, her bir tik üzerinde göstergenin zorla hesaplanmasını sağlayabilirsiniz. Aşağıdaki durumlarda da Strateji Sınayıcısındaki göstergeler her bir tik üzerinde zorla hesaplanır: • görsel modda test edilirken; • gösterge aşağıdaki fonksiyonlardan birine sahipse: EventChartCustom, OnChartEvent, OnTimer; • eğer gösterge 1916'dan aşağı bir yapı numarasına sahip bir derleyici ile oluşturulduysa. Bu özellik sadece Strateji Sınayıcısında geçerlidir; terminal de ise göstergeler her zaman, alınan her bir tik üzerinde hesaplanır.
optimization_chart_mode	string	Grafik türünü ve optimizasyon sonuçlarının görselleştirilmesi için kullanılacak iki girdi parametresinin adını belirtir. Örneğin, "3d, InpX, InpY"; sonuçların, test edilen InpX ve InpY parametre değerlerini temel alan koordinat eksenlerine sahip bir 3D grafikte gösterileceği anlamına gelir. Böylece, bu özelliği kullanarak, optimizasyon grafiğini ve grafik türünü program kodunda görüntülemek için kullanılacak parametreleri doğrudan belirleyebilirsiniz. Olası seçenekler: • "3d, input_parameter_name1, input_parameter_name2"; döndürülebilen, yakınlaştırılabilen ve uzaklaştırılabilen bir 3D görselleştirme grafiği anlamına gelir. Grafik iki parametre kullanılarak oluşturulur. • "2d, input_parameter_name1, input_parameter_name2", sonuçlara bağlı olarak her bir hücrenin belirli bir renkte boyandığı bir 2D ızgara grafik anlamına gelir. Grafik iki parametre kullanılarak oluşturulur. • "1d, input_parameter_name1, input_parameter_name2", sonuçların belirtilen parametreye göre sıralandığı doğrusal bir grafik anlamına gelir. Her geçiş

Sabit	Tip	Açıklama
		bir nokta olarak gösterilir. Grafik tek parametreye dayanılarak oluşturulur. • "0d, input_parameter_name1, input_parameter_name2", sonuçların geçiş sonucunun varış zamanına göre sıralandığı normal bir grafik anlamına gelir. Her geçiş grafikte bir nokta olarak gösterilir. Parametre belirtilmesi gerekli değildir, ancak belirtilen parametreler diğer grafik türlerine manuel geçiş için kullanılabilir. İsteğe bağlı olarak, bir veya iki giriş parametresi belirtmeden yalnızca grafik türünü belirtebilirsiniz. Bu durumda; terminal, optimizasyon grafiğini göstermek için gerekli parametreleri seçecektir.

Açıklama ve Versiyon Numarası için Örnek

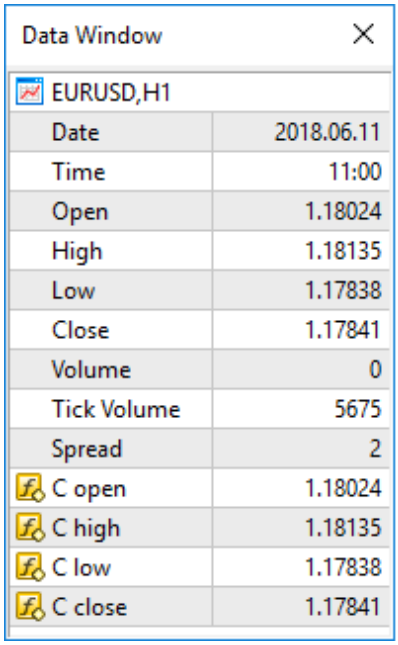
```
#property version     "3.70"      // Uzman Danışmanın mevcut versiyonu
#property description "Pesavento Motifleriyle Evrensel ZigZag"
#property description "Şu anda göstergede farklı algoritmalarla birkaç zigzag içerilme
#property description "Yüksek ve düşük değerleri gösteren ve bunlardan otomatik olarak
#property description "araçlar inşa eden çok sayıda başka göstergenin de, buna eklenme
```



Her bir Gösterge Tamponu için Ayrı bir Etiket Belirleme Örneği ("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"
```



Dosya Ekleme (#include)

#include komutu program içinde her hangi bir konuma yerleştirilebilir ama genellikle tüm ek dosyalar kaynak kodunun başlangıcında yazılır. Çağrı biçimi:

```
#include <dosya_ismi>
#include "dosya_ismi"
```

Örnekler:

```
#include <WinUser32.mqh>
#include "mylib.mqh"
```

Önişlemci *#include <dosya_ismi>* satırının yerini, WinUser32.mqh dosyasının içeriği ile değiştirir. Açı parantezler WinUser32.mqh dosyasının standart dosya konumundan alınacağını gösterir (genellikle şöyledir: *terminal_yükleme_konumu\MQL5\Include*). Mevcut dosya yolu görünür değildir.

Dosya ismi tırnak içinde verilmişse, arama mevcut dosya konumunda (ana kaynak dosyasını içeren konumda) yapılır. Standart kütüphane dizini görünür değildir.

Ayrıca Bakınız

Standart Kütüphane, Fonksiyonların İçe Aktarımı

Fonksiyonun İçe Aktarımı (#import)

Fonksiyonlar, derlenmiş MQL5 modüllerinden (*.ex5 dosyaları) ve işletim sistemi modüllerinden (*.dll dosyaları) içe aktarılır. Modülün ismi *#import* direktifinde belirtilir. Derleyicinin, fonksiyon çağrısını ve aktarılan parametreleri doğru şekilde biçimlendirebilmesi için, fonksiyonların tam tarifi gereklidir. Fonksiyon tarifleri *#import "modül ismi"* direktifinin hemen ardından gelir. Yeni komut *#import* (parametresiz olabilir), içe aktarılan fonksiyonların tariflerinden oluşan bloğu tamamlar.

```
#import "file_name"
   fonk1 tanımla;
   fonk2 tanımla;
   ...
   fonkN tanımla;
#import
```

İçe aktarılan fonksiyonlar herhangi bir isme sahip olabilir. Aynı isme ama farklı modüllere sahip iki fonksiyon aynı anda tanımlanabilir. İçe aktarılan fonksiyonların isimleri, gömülü fonksiyon isimleriyle aynı olabilir. Kapsam çözünürlüğü işlemi, hangi fonksiyonların çağrılması gerektiğini tanımlar.

#import anahtar sözcüğünden sonra belirlenen dosya arama emri İçe Aktarılan Fonksiyonların Çağrısı içinde tanımlanır.

İçe aktarılan fonksiyonlar derlenen modülün içinde olmadığından, derleyici geçirilen parametreleri doğrulayamaz. Bu yüzden, çalışma zamanı hatalarından kaçınmak amacıyla, içe aktarılan fonksiyona geçirilen parametrelerin düzeni ve sırası doğru tarif edilmelidir. İçe aktarılan (EX5'den veya DLL modülünden) fonksiyonlara geçirilen parametreler ön tanımlı değerlere sahip olabilirler.

İçe aktarılan fonksiyonlarda şunlar kullanılamaz:

- işaretçiler (*);
- Dinamik dizilere ve/veya işaretçilere verilen linkler.

Dizgiler veya herhangi tipli dinamik diziler içeren sınıflar, string tipi diziler veya karmaşık nesneler, DLL dosyasından içe aktarılan fonksiyonlara parametre olarak geçirilemezler

Örnekler:

```
#import "stdlib.ex5"
string ErrorDescription(int error_code);
int    RGB(int red_value,int green_value,int blue_value);
bool   CompareDoubles(double number1,double number2);
string DoubleToStrMorePrecision(double number,int precision);
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
```

MQL5 programının çalıştırılması sırasında fonksiyonların içe aktarımı için erken bağlama kullanılır. Yani kütüphaneler, programın ex5 uzantılı dosyası ile yüklenir.

Yüklenebilir modülünün bütünüyle belirtilmiş isminin (*Sürücü:\Dizin\Dosyaismi.Uzn*) kullanılması tavsiye edilmez. MQL5 kütüphaneleri *terminal_konumu\MQL5\Libraries* dosyasından yüklenir.

İçe aktarılan işlev 32 ve 64 bit Windows sürümleri için farklı çağrı sürümlerine sahipse, her ikisi de içe aktarılmalı ve _IsX64 değişkenini kullanarak doğru işlev sürümü açıkça çağrılmalıdır.

Örnek:

```
#import "user32.dll"
//--- 32-bit sistem için
int    MessageBoxW(uint hWnd,string lpText,string lpCaption,uint uType);
//--- 64-bit sistem için
int    MessageBoxW(ulong hWnd,string lpText,string lpCaption,uint uType);
#import
//+------------------------------------------------------------------+
//|  MessageBox_32_64_bit MessageBoxW()'un uygun sürümünü kullanır   |
//+------------------------------------------------------------------+
int MessageBox_32_64_bit()
  {
   int res=-1;
   //--- Eğer 64-bit Windows kullanıyorsak
   if(_IsX64)
     {
      ulong hwnd=0;
      res=MessageBoxW(hwnd,"64-bit MessageBoxW call example","MessageBoxW 64 bit",MB_
     }
   else  // 32-bit Windows kullanıyoruz.
     {
      uint hwnd=0;
      res=MessageBoxW(hwnd,"32-bit MessageBoxW call example","MessageBoxW 32 bit",MB_
     }
   return (res);
  }
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int ans=MessageBox_32_64_bit();
   PrintFormat("MessageBox_32_64_bit returned %d",ans);
  }
```

.NET kütüphanelerinden fonksiyonları içe aktarma

.NET kütüphane fonksiyonlarıyla çalışmak için, belirli fonksiyonları tanımlamadan DLL'nin kendisini içe aktarmanız yeterlidir. MetaEditor, çalışabileceğiniz tüm fonksiyonları otomatik olarak içe aktarır:

- Basit yapılar (POD, plain old data) - yalnızca basit veri tiplerini içeren yapılar.
- Sadece basit tiplerin ve POD yapılarının veya dizilerinin kullanıldığı parametrelere sahip 'public static' fonksiyonlar.

Kütüphaneden fonksiyonları çağırmak için içe aktarmanız yeterlidir:

```
#import "TestLib.dll"
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   int x=41;
   TestClass::Inc(x);
   Print(x);
  }
```

TestClass sınıfının Inc fonksiyonunun C# kodu aşağıdaki gibidir:

```
public class TestClass
  {
   public static void Inc(ref int x)
     {
      x++;
     }
  }
```

Yürütme sonucunda, komut dosyası 42 değerini geri döndürür.

Ayrıca Bakınız

Dosya Ekleme

Koşullu Derleme (#ifdef, #ifndef, #else, #endif)

Önişlemci yönergeleri derleyici tarafından derlenmeden önce kaynak kodunu ön işleme koymak için kullanılır. Yönerge her zaman # ile başlar, bu nedenle derleyici değişkenlerin, fonksiyonların vb. simlerinde sembol kullanımını yasaklar.

Her direktif ayrı bir girdi ile tanımlanır ve satır sonuna kadar geçerlidir. Bir girişte birkaç yönergeyi kullanamazsınız. Yönerge girişi çok büyükse, '\' simgesini kullanarak birkaç satıra bölünebilir. Bu durumda, bir sonraki satır direktif girişinin bir devamı olarak kabul edilir.

Önişlemci koşullu derleme direktifleri, belirli bir koşulun gerçekleşmesine bağlı olarak, programın bir bölümünü derlemenizi veya atlamnızı sağlar.

Bahsedilen bu koşul aşağıdaki şekillerden birini alabilir.

```
#ifdef identifier
   // tanımlayıcı, #define direktifindeki ön işlemci için tanımlanmışsa, burada yer al
#endif
```

```
#ifndef tanımlayıcı
   // tanımlayıcı henüz #define önişlemci direktifi tarafından tanımlanmamışsa burada
#endif
```

Koşullu derleme direktiflerinin tümü, muhtemelen #else direktifini taşıyan ve #endif ile sonlanan herhangi sayıda satır ile devam edebilir. Onaylanan koşulun doğru olması durumunda #else ve #endif arasında kalan satırlar gözardı edilir. Onaylanan koşul gerçekleşmemişse, kontrol ile #else direktifi (veya biçimlendiricinin olmaması durumunda #endif direktifi) arasında kalan tüm satırlar gözardı edilir.

Örnek:

```
#ifndef TestMode
   #define TestMode
#endif
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   #ifdef TestMode
      Print("Sınama modu");
   #else
      Print("Normal mod");
   #endif
  }
```

Standart makrolar, program tipine ve derleme moduna bağlı olarak şu şekilde tanımlanırlar:

__MQL5__ makrosu bir *.mq5 dosyası derlenirken, __MQL4__ makrosu ise bir *.mq4 dosyası derlenirken tanımlanır.
_DEBUG makrosu, hata ayıklama modunda,
_RELEASE makrosu ise sürüm modunda derleme yapılırken tanımlanır.

Örnek:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
#ifdef __MQL5__
   #ifdef _DEBUG
      Print("MQL5 derleyicisinden merhaba [DEBUG]");
   #else
      #ifdef _RELEASE
         Print("MQL5 derleyicisinden merhaba [RELEASE]");
      #endif
   #endif
#else
   #ifdef __MQL4__
      #ifdef _DEBUG
         Print("MQL4 derleyicisinden merhaba [DEBUG]");
      #else
         #ifdef _RELEASE
            Print("MQL4 derleyicisinden merhaba [RELEASE]");
         #endif
      #endif
   #endif
#endif
  }
```

Nesne Yönelimli Programlama

Nesne yönelimli programlama (NYP), verilerin ve davranışların ayrılmaz şekilde bağlı olduğu bir süreçte, öncelikli olarak veriye odaklanmış bir programlamadır. Veri ve davranış birlikte bir sınıfı meydana getirirler ve nesneler de sınıf oluşumlarıdır.

Nesne yönelimli yaklaşımın bileşenleri şunlardır:

- Kapsülleme ve tiplerin genişletilebilirliği
- Kalıtım
- Polimorfizm
- Aşırı-yükleme
- Sanal fonksiyonlar

NYP hesaplamayı davranış modellemesi olarak görür. Modellenen parça, hesaba dayalı soyutlamalar aracılığıyla temsil edilen nesnedir. Oldukça iyi bilinen "Tetris" oyununu yazmak istediğimizi varsayalım. Bunu yapmak için, kenarlarından birbirine bağlanmış dört kareden oluşan rassal şekillerin görünümünü nasıl modelleyeceğimizi öğrenmeliyiz. Ayrıca, şekillerin düşüş hızlarını düzenlemeli, yönlendirme ve kaydırma işlemlerini tanımlamalıyız. Şekillerin ekrandaki hareketleri kuyunun sınırlarıyla belirlendiğinden bu gereksinimi de modellememiz gerekir. Bunun yanında, küplerle doldurulmuş satırlar yok edilmeli ve kazanılmış puanlar işlenmelidir.

Kısacası bu anlaşılması kolay oyun, çeşitli modellerin (şekil modeli, kuyu modeli, şekil hareketi modeli ve dahası) oluşturulmasını gerektirir. Tüm bu modeller soyutlama olarak adlandırılır ve bilgisayardaki hesaplamalar tarafından temsil edilirler. Bunları tanımlamak için Soyut Veri Tipi, SVT (veya karmaşık veri tipi) kullanılır. Aslında, kuyu içindeki "şekil" hareketinin modeli bir veri tipi değil, "şekil" veri tipi üzerindeki bir işlem kümesidir ve "kuyu" veri tipinin kısıtlamalarını kullanır.

Nesneler sınıf değişkenleridir. Nesne yönelimli programlama sayesinde bir SVT oluşturup kullanmak oldukça kolaydır. Nesne yönelimli programlama kalıtım mekanizmasını kullanır. Kalıtım, kullanıcı tarafından zaten oluşturulmuş olan veri tiplerinin, türevlerinin elde edilmesini sağlar.

Örneğin, Tetris şekillerini oluşturmak için önce bir temel Şekil sınıfı oluşturmak uygun olacaktır. Yedi olası şeklin tamamını temsil eden diğer sınıflar bu temel sınıftan türetilebilir. Şekillerin davranışları temel sınıfta tanımlanırken, bu davranışların ayrı şekiller tarafından uygulanması türetik sınıflarda tanımlanır.

NYP'da nesneler kendi davranışlarından sorumludur. SVT geliştiricisinin, nesneden normalde beklenebilecek tüm davranışları tarif etmek için bir kodu olmalıdır. Nesnenin kendi davranışlarından sorumlu olduğu gerçeği, nesne kullanıcısının programlama görevini büyük oranda kolaylaştırır.

Ekrana bir şekil çizmek istediğimizde merkezin neresi olacağını ve nasıl çizeceğimizi bilmemiz gerekir. Eğer bir şekil kendisini nasıl çizeceğini biliyorsa, programcının bunu kullanırken bir "çiz" mesajı göndermesi gerekir.

MQL5 dili, C++ benzeri bir dildir ve SVT uygulamaları için kapsülleme mekanizmasına da sahiptir. Kapsülleme bir taraftan özel bir tipin uygulamasındaki iç detayları birleştirirken, diğer taraftan, bu tipteki nesneleri etkileyebilecek, dışarıdan erişilebilen fonksiyonları birleştirir. Bu tipi kullanan bir program için uygulama detayları erişilmez olabilir.

NYP konsepti aşağıda verilenler gibi bir takım ilişkili kavramlar içerir:

- Gerçek dünyadan olayların simülasyonu
- Kullanıcı tanımlı veri tipleri
- Uygulama detaylarının gizlenmesi
- Kalıtım yoluyla kodun tekrar kullanılma olasılığı
- Çalıştırma esnasında fonksiyon çağrılarının açıklanması

Bu kavramların bir kısmı oldukça belirsizken, bazıları soyut, diğerleri ise geneldir.

Tiplerin Kapsüllenmesi ve Genişletilebilirliği

NYP yazılım üretmek için dengelenmiş bir yaklaşımdır. Veri ve davranış birlikte paketlenir. Bu kapsülleme, kullanıcı tanımlı veri tipleri oluşturur, dildeki veri tiplerini genişletir ve onlarla etkileşim kurar. Tiplerin genişletilebilirliği, dile kullanıcı tanımlı yeni tipler kazandırır. Bu yeni veri tipleri de en az temel tipler kadar kolay kullanıma sahiptirler.

Bir soyut veri tipi (örneğin bir dizgi) ideal ve iyi bilinen bir davranış tipinin tarifidir.

Dizgi kullanıcısı birleştirme veya yazma gibi dizgi işlemlerini bilir. Bu tür işlemler yöntem olarak isimlendirilir.

SVT ile yapılan belli uygulamalar bir takım kısıtlamalara tabi olabilir. Örneğin, dizgilerin uzunlukları sınırlıdır. Sınırlamalar her şeye açık olan davranışı etkiler. Aynı zamanda, içsel veya özel uygulama detayları kullanıcının nesneyi görme şeklini doğrudan etkilemez. Örneğin, dahili taban adresleri ve isim kullanıcı için gerekli değilse, dizgi çoğu zaman bir dizi gibi uygulanır.

Kapsülleme, kullanıcı tanımlı tipler için açık arayüzler temin edildiğinde, uygulama detaylarını saklamaya yarayan bir yetenektir. MQL5 'da, tıpkı C++ 'da olduğu gibi, sınıf (class) ve yapı (struct) tanımları, anahtar erişim belirteçleriyle (private, protected ve public) kombine şekilde, kapsülleme provizyonları için kullanılır.

public (özel) anahtar sözcüğü kendinden sonra gelen üyeler için kısıtlamasız, açık bir erişim ifade eder. Bu anahtar sözcük olmadan sınıf üyeleri varsayılan olarak kilitli durumdadırlar. Özel üyeler sadece kendi sınıfları içinde yer alan üye fonksiyonlar için erişilebilir durumdadır.

Korunan (protected) sınıf fonksiyonları sadece kendi sınıfları için değil, aynı zamanda kalıtımsal türev sınıflar içinde kullanılabilir olacaktır. Public (genele açık) sınıf fonksiyonları, sınıf bildirimi çerçevesindeki tüm fonksiyonlar için kullanılabilir durumdadır. Koruma, sınıfın uygulama bölümünün gizlenmesine ve bu sayede veri yapısındaki değişiklerin önlenmesine olanak sağlar. Erişim kısıtlaması veya veri gizleme nesne yönelimli programlamanın bir özelliğidir.

Sınıf fonksiyonları genellikle protected şekillendiricisi ile korunarak bildirilir. Değerlerin okunup yazılması ise, ayarla-ve-al şeklinde isimlendirilen yöntemlerle gerçekleştirilir. Bu yöntemler public erişim şekillendiricisi ile belirlenirler.

Örnek:

```
class CPerson
  {
protected:
   string            m_name;                    // isim
public:
   void              SetName(string n){m_name=n;}// ismi ayarlar
   string            GetName(){return (m_name);} // isme dönüş yapar
  };
```

Bu yaklaşım bize çeşitli avantajlar sağlar. İlk olarak, isim aracılığıyla fonksiyonun ne işe yaradığını anlarız - sınıf üyesinin değerini ayarlar veya alır. İkinci olarak, gelecekte CPerson sınıfının -veya bunun türev sınıflarının içindeki m_name değişkeninin tipini- değiştirmemiz gerekebilir.

Bu durumda, CPerson sınıfının nesneleri, hiçbir kod değişimi olmadan programda kullanılabilir olacağından, sadece SetName() ve GetName() fonksiyonlarının uygulama şeklini değiştirmemiz yeterli olur. Çünkü kullanıcı m_name'in veri tipinin değiştiğini bile bilmeyecektir.

Örnek:

```
struct Name
  {
   string            first_name;                  // isim
   string            last_name;                   // soy isim
  };

class CPerson
  {
protected:
   Name              m_name;                      // isim
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);
  }
```

Ayrıca Bakınız

Veri Tipleri

Kalıtım

Kodun kalıtım yoluyla yeniden kullanımının teşvik edilmesi NYP 'nin karakteristik özelliğidir. Temel sınıf olarak adlandırılan bir sınıftan yeni bir sınıf elde edilir. Türetilen sınıf, temel sınıfın üyelerini kullanır ama bunlar üzerinde şekillendirmeler ve ilaveler de yapabilir.

Birçok tip zaten var olan başka bir tipin versiyonudur. Bunların her biri için yeni bir kod yazmak çoğunlukla yorucudur. Ayrıca, yeni kod yeni hatalar anlamına gelir. Türetik sınıf temel sınıfın tarifini miras alır. Bu şekilde kodun yeniden geliştirilmesi ve sınanması gereksiz olur. Kalıtım ilişkileri hiyerarşiktir.

Hiyerarşi, elemanların tüm çeşitlilik ve karmaşıklığı ile kopyalanmasını sağlayan bir yöntemdir. Nesnelerin sınıflandırmasını sağlar. Örneğin, elementlerin periyodik tablosunda gazlar vardır. Bunlar tüm elementlerde kalıtımsal olan özellikleri taşırlar.

Asal gazlar ise bir sonraki önemli alt sınıfı oluşturur. Burada hiyerarşi, argon gibi bir asal gazın, bir gaz olması ve gazın da kendi çapında sistemin bir parçası olmasıdır. Böylece, asal gazın davranışının kolayca açıklanması sağlanır. Bunların atomlarının, protonları ve elektronları içerdiğini biliriz -ki bu diğer tüm elementler için de geçerlidir.

Diğer tüm gazlar gibi, oda sıcaklığında gaz halinde olduklarını biliriz. Asal gaz alt sınıfından hiç bir gazın diğer elementlerle kimyasal reaksiyona girmediğini biliriz; bu bütün asal gazların özelliğidir.

Geometrik şekillerin kalıtımına dair bir örneği göz önüne getirelim. Basit şekillerden (daire, üçgen, dikdörtgen, kare vb.) oluşan bir grubu tanımlamak için en iyi yol , her bir türetik şeklin atası olan bir temel sınıf (SVT) oluşturmaktır.

Şekli tarif eden en bilindik üyeleri içeren CShape isminde bir temel sınıf oluşturalım. Bu üyeler tüm şekillerde karakteristik olan özellikleri (şekil tipi ve tutturma koordinatları) tarif eder.

Örnek:

```
//--- Şekil temel sınıfı
class CShape
  {
protected:
   int       m_type;                    // Şekil tipi
   int       m_xpos;                    // X - merkez nokta koordinatı
   int       m_ypos;                    // Y - merkez nokta koordinatı
public:
             CShape(){m_type=0; m_xpos=0; m_ypos=0;} // yapıcı
   void      SetXPos(int x){m_xpos=x;} // X'i ayarla
   void      SetYPos(int y){m_ypos=y;} // Y'yi ayarla
  };
```

Daha sonra, temel sınıftan türetilen yeni sınıflar oluşturulur. Bunlara, her biri belli bir sınıfı tanımlayan gerekli alanlar eklenir. Daire şekli için, yarıçap değerini içeren bir üyenin eklenmesi gereklidir. Kare şekli ise kenar değeri ile nitelenir. Bu yüzden, CShape temel sınıfından kalıtım yoluyla türetilen alt sınıflar, şu şekilde bildirilirler:

```
//--- Türetik sınıf daire
class CCircle : public CShape         // İki nokta işaretinden sonra temel sınıfı tanır
```

```
  {                                    // kalıtımın geldiği sınıf
private:
   int             m_radius;           // daire yarıçapı

public:
                   CCircle(){m_type=1;}// yapıcı, tip 1
  };
```

Şekil sınıfının bildirimi Kare için de benzerdir:

```
//--- türetik sınıf Kare
class CSquare : public CShape        // İki nokta işaretinden sonra temel sınıfı tanır
  {                                    // kalıtımın geldiği sınıf
private:
   int             m_square_side;      // kare kenarı

public:
                   CSquare(){m_type=2;} // yapıcı, tip 2
  };
```

Nesne oluşturulurken ilk olarak temel sınıf yapıcısının, daha sonra ise türetik sınıfın yapıcısının çağrılacağı not edilmelidir. Nesne yok edilirken ise, önce türev sınıf yıkıcısı, ardından temel sınıf yıkıcısı çağrılır.

Bu şekilde, en genel üyeleri temel sınıfta bildirip, daha özel sınıfları belirleyen diğer üyeleri türetik sınıflara ekleyebiliriz. Kalıtım defalarca kullanılabilen güçlü kodların oluşturulmasını sağlar.

Halihazırda mevcut olan bir sınıftan, bir türetik sınıf oluşturulmasının sözdizimi şu şekildedir:

```
class sınıf_ismi :
          (public | protected | private) opt  temel_sınıf_ismi
  {
   sınıf üyelerinin bildirimi
  };
```

Türetik sınıfın önemli bir yönü, üyelerinin ve haleflerinin (kalıtımsal varisler) görünürlüğüdür. 'public', 'protected' ve 'private' erişim belirteçleri, kapsamı - yani temel sınıfın hangi üyelerinin, türetik sınıflar için mevcut olacağını - belirtmek için kullanılır. Türetik sınıfın başlığında iki nokta işaretinden sonra yer alan 'public' anahtar sözcüğü, CShape temel sınıfının korunan (protected) ve genele açık (public) üyelerinin, CCircle türetik sınıfının korunan ve genele açık üyeleri şeklinde miras alınması gerektiğini belirtir.

Temel sınıfın özel (private) üyeleri, türetik sınıf için kullanılabilir değildir. Genele açık kalıtım, türetik sınıfların (CCircle ve CSquare) aynı zamanda CShapes sınıfı oldukları anlamına da gelir. Şöyle ki, Kare (CSquare) bir şekildir (CShape) ama şekil kare olmak zorunda değildir.

Türetik sınıf, temel sınıfın bir modifikasyonudur ve temel sınıfın korunan ve genele açık üyelerini miras alır. Temel sınıfın yapıcıları ve yıkıcıları miras bırakılamaz. Temel sınıfın üyelerine ek olarak, yeni üyeler de türetik sınıfa eklenir.

Türetik sınıf, üye fonksiyonların temel sınıftan farklı uygulamalarını içerebilir. Bunun aşırı-yükleme ile herhangi bir alakası yoktur; çünkü aynı fonksiyon ismi farklı imzalarla kullanılabilir.

Korunan kalıtım durumunda, temel sınıfın genele açık ve korunan üyeleri, türetik sınıfın korunan üyeleri haline gelir. Özel kalıtım durumunda, temel sınıfın genele açık ve korunan üyeleri, türetik sınıfın özel (private) üyeleri haline gelir.

Korunan ve özel kalıtımlarda, "bir türetik sınıf nesnesi aynı zamanda bir temel sınıf nesnesidir" ilişkisi doğru değildir. Korunan ve özel kalıtım tipleri nadirdir ve her biri dikkatle kullanılmalıdır.

Kalıtımsal hiyerarşi ile türetik sınıftan temel sınıfın üyelerine gerçekleştirilen erişim şekillerinin, kalıtım tipinden (public, protected veya private) etkilenmeyeceği not edilmelidir. Tüm kalıtım türlerinde, sadece public ve protected erişim belirteçleri ile bildirilen temel sınıf üyeleri türetik sınıfın dışında kullanılabilir. Şu örneği göz önüne alalım:

```
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//+------------------------------------------------------------------+
//| Birkaç erişim tipine sahip sınıf örneği                          |
//+------------------------------------------------------------------+
class CBaseClass
  {
private:             //--- Özel üye, türetik sınıflar için mevcut değil
   int               m_member;
protected:           //--- Korunan yöntem temel sınıf ve türetik sınıflar için mevcut
   int               Member(){return(m_member);}
public:              //--- Sınıf yapıcısı tüm sınıfların üyeleri için mevcut durumda
                     CBaseClass(){m_member=5;return;};
private:             //--- m_member değişkenine değer atamak için özel bir yöntem
   void              Member(int value) { m_member=value;};

  };
//+------------------------------------------------------------------+
//| Hatalarla türetik sınıf                                          |
//+------------------------------------------------------------------+
class CDerived: public CBaseClass // genele açık kalıtımın tanımı (ön tanımlıysa) dahi
  {
public:
   void Func() // türetik sınıfta, temel sınıf üyelerinin çağrılarını içeren bir fonks
     {
      //--- temel sınıfın özel bir üyesini şekillendirme denemesi
      m_member=0;         // Hata; temel sınıfın özel üyesi mevcut değil
      Member(0);          // Hata; temel sınıfın özel yöntemi, türetik sınıflarda mevcu
      //--- Temel sınıf üyesinin okunması
      Print(m_member);    // Hata; temel sınıfın özel üyesi mevcut değil
      Print(Member());    // Hata yok; korunan yöntem, temel sınıfta ve türetik sınıfla
     }
  };
```

Yukarıdaki örnekte, CBaseClass sadece genele açık (public) yönteme sahip - yapıcı. Yapıcılar bir sınıf nesnesi oluşturulduğunda otomatik olarak çağrılırlar. Bu yüzden, özel üye m_member ve korunan yöntemler Member() dışarıdan çağrılamazlar. Ama genele açık kalıtım durumunda, temel sınıfın Member() yöntemi türetik sınıflardan kullanılabilir olacaktır.

Korunan kalıtım durumunda ise, public ve protected belirteçlerine sahip olan tüm temel sınıf üyeleri korunur. Bu demektir ki; temel sınıfın genele açık veri üyelerine ve yöntemlerine dışarıdan erişim serbestken, korunan kalıtım durumunda sadece türetik sınıftan ve onun türevlerinden erişim yapılabilir.

```
//+------------------------------------------------------------------+
//| Birkaç erişim tipine sahip sınıf örneği                          |
//+------------------------------------------------------------------+
class CBaseMathClass
  {
private:             //--- Özel üye, türetik sınıflar için mevcut değil
   double            m_Pi;
public:              //--- m_Pi için bir değerin alınıp ayarlanması
   void              SetPI(double v){m_Pi=v;return;};
   double            GetPI(){return m_Pi;};
public:              // Sınıf yapıcısı tüm üyelerce kullanılabilir
                     CBaseMathClass() {SetPI(3.14);  PrintFormat("%s",__FUNCTION__);};
  };
//+------------------------------------------------------------------+
//| m_Pi'nin değiştirilemeyeceği bir türetik sınıf                   |
//+------------------------------------------------------------------+
class CProtectedChildClass: protected CBaseMathClass // Korunan kalıtım
  {
private:
   double            m_radius;
public:              //--- Türetik sınıftaki genele açık yöntemler
   void              SetRadius(double r){m_radius=r; return;};
   double            GetCircleLength(){return GetPI()*m_radius;};
  };
//+------------------------------------------------------------------+
//| Script starting function                                         |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Türetik sınıf oluşturulurken, temel sınıfın yapıcısı otomatik olarak çağrılır
   CProtectedChildClass pt;
//--- Yarı çapı belirle
   pt.SetRadius(10);
   PrintFormat("Length=%G",pt.GetCircleLength());
//--- Eğer dizgi aşağıdaki gibi yorumlanırsa, SetPi() korunduğu için derleme aşamasınd
// pt.SetPI(3);

//--- Şimdi temel sınıfın bir değişkenini bildirelim ve Pi sabitini 10 olarak ayarlaya
   CBaseMathClass bc;
```

```
   bc.SetPI(10);
//--- Sonuç şöyle olur
   PrintFormat("bc.GetPI()=%G",bc.GetPI());
  }
```

Örneğe göre, CBaseMathClass temel sınıfındaki SetPI() ve GetPi() yöntemleri açık erişime sahiptir ve programın herhangi bir yerinden çağrılabilirler. Ama aynı zamanda türetik CProtectedChildClass için, bu yöntemler sadece CProtectedChildClass sınıfının veya onun türetik sınıflarının yöntemlerinden çağrılabilir.

Özel kalıtım durumunda, temel sınıfın genele açıkve korunan erişime sahip tüm üyeleri özel hale gelir; daha sonraki kalıtımlarda çağrılmaları imkansız olur.

MQL5 çoklu kalıtım içermez.

Ayrıca Bakınız

Yapılar ve Sınıflar

Polimorfizm

Polimorfizm, aynı fonksiyon elemanının çağrılması durumunda farklı cevaplar alınabildiği için, kalıtımsal ilişkiye sahip faklı nesne sınıfları için bir fırsattır. Bu, sadece temel sınıfın değil aynı zamanda türetik sınıfların davranışlarının da tarif edilebildiği evrensel bir mekanizma oluşturmaya yardımcı olur.

Şimdi yeni bir temel sınıf oluşturalım ve şekil alanını hesaplaması için GetArea() diye bir üye fonksiyon tanımlayalım. Bu fonksiyonu, temel sınıftan kalıtım yoluyla oluşturulmuş tüm varis sınıflarda, belirlenen özel şeklin alan hesabı için gerekli kurallar ışığında, yeniden tanımlayalım.

Karenin (CSquare sınıfı) alanı kenarlara göre hesaplanır;, Daire (class CCircle) alanı ise yarı çapla ifade edilir vb. CShape tipinin nesnelerini içeren bir dizi oluşturabiliriz. Bu, temel sınıfın ve tüm varis sınıfların nesnelerinin saklandığı bir dizi olacaktır. Sonra bu dizinin her elemanı için aynı fonksiyonu çağırabiliriz.

Örnek:

```
//--- Temel sınıf
class CShape
  {
protected:
   int            m_type;                // Şekil tipi
   int            m_xpos;                // merkez noktanın X - koordinatları
   int            m_ypos;                // merkez noktanın Y - koordinatları
public:
   void           CShape(){m_type=0;};   // yapıcı, type=0
   int            GetType(){return(m_type);};// nesne tipine dönüş yapar
virtual
   double         GetArea(){return (0); }// şeklin alanına dönüş yapar
  };
```

Artık tüm türetik sınıflarda sıfıra dönüş yapan bir getArea() üye fonksiyonu vardır. Bu fonksiyonun uygulaması her bir türetik sınıf için değişecektir.

```
//--- Daire türetik sınıfı
class CCircle : public CShape             // İki noktadan sonra temel sınıfı tanımlarız
  {                                       // -ki bu, kalıtımın yapıldığı sınıftır
private:
   double         m_radius;              // daire yarı çapı

public:
   void           CCircle(){m_type=1;};  // yapıcı, type=1
   void           SetRadius(double r){m_radius=r;};
   virtual double GetArea(){return (3.14*m_radius*m_radius);}// daire alanı
  };
```

Kare sınıfı için yapılacak bildirim de aynıdır:

```
//--- Kare türetik sınıfı
class CSquare : public CShape             // İki noktadan sonra temel sınıfı tanımlarız
  {                                       // -ki bu, kalıtımın yapıldığı sınıftır
```

```
private:
   double           m_square_side;         // kare kenarı

public:
   void             CSquare(){m_type=2;}; // yapıcı, type=1
   void             SetSide(double s){m_square_side=s;};
   virtual double   GetArea(){return (m_square_side*m_square_side);}// kare alanı
  };
```

Kare ve dairenin alanlarını hesaplamak için m_radius (yarı çap) ve m_square_side (kenar uzunluğu) değişkenlerine karşılık gelen değerlere ihtiyacımız oldu için, SetRadius() ve SetSide() fonksiyonlarını bildirime ekledik.

CShape temel tipinden türetilmiş (CCircle ve CSquare) nesnelerinin, programımızda kullanıldığını varsayalım. Polimorfizm, CShape temel sınıfının nesnelerinden bir dizi oluşturulmasını sağlar ama bu dizinin bildirimi sırasında nesneler hala bilinmezdir ve tipleri tanımsızdır.

Dizinin her bir elemanında hangi tip nesnenin yer alacağı kararı programın çalıştırılması sırasında alınır. Bu, uygun sınıfların nesnelerinin dinamik olarak oluşturulmasını ve bundan dolayı, nesneler yerine nesne işaretçilerinin kullanımının gerekliliğini içerir.

new operatörü nesnelerin dinamik şekilde oluşturulmaları için kullanılır. Bu şekildeki her bir nesne delete operatörü ile, tek tek ve açıkça silinmelidir. Bu yüzden CShape tipinin işaretçilerinden oluşan bir dizi bildireceğiz ve aşağıdaki örnekte de görüleceği gibi, her bir **(new Sınıf_İsmi)** eleman için uygun tipte bir nesne oluşturacağız:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Temel tipin nesne işaretçilerinden oluşan bir dizi bildir
   CShape *shapes[5];   // CShape nesnesinin işaretçilerinden oluşan dizi

//--- Burada diziyi türetilen nesnelerle doldur
//--- CCircle tipi nesneler için bir işaretçi bildir
   CCircle *circle=new CCircle();
//--- Daire işaretçisinde nesne özelliklerini ayarla
   circle.SetRadius(2.5);
//--- İşaretçi değerini shapes[0] içine yerleştir
   shapes[0]=circle;

//--- Yeni bir CCircle nesnesi oluştur ve işaretçisini shapes[1] içine yaz
   circle=new CCircle();
   shapes[1]=circle;
   circle.SetRadius(5);

//--- Burada shapes[2] için değer ayarlamayı bilerek unuttuk
//circle=new CCircle();
//circle.SetRadius(10);
//shapes[2]=circle;
```

```
//--- Kullanılmayan elemanları NULL (boş) olarak ayarla
   shapes[2]=NULL;

//--- Yeni bir CSquare nesnesi oluştur ve işaretçisini shapes[3] içine yaz
   CSquare *square=new CSquare();
   square.SetSide(5);
   shapes[3]=square;

//--- Yeni bir CSquare nesnesi oluştur ve işaretçisini shapes[4] içine yaz
   square=new CSquare();
   square.SetSide(10);
   shapes[4]=square;

//--- İşaretçilerden oluşan bir dizimiz var bunun büyüklüğünü al
   int total=ArraySize(shapes);
//--- Dizideki tüm işaretçileri bir döngüye geçir
   for(int i=0; i<5;i++)
     {
      //--- Eğer belirli bir indisteki işaretçi geçerliyse
      if(CheckPointer(shapes[i])!=POINTER_INVALID)
        {
         //--- Şeklin tipini günlüğe kaydet
         PrintFormat("%d tipli nesne, %G karesine sahip",
               shapes[i].GetType(),
               shapes[i].GetArea());
        }
      //--- Eğer işaretçi POINTER_INVALID tipinde ise
      else
        {
         //--- Bir hatay ile uyar
         PrintFormat("Nesne shapes[%d] başlatılmadı! Bu nesnenin işaretçişi %s",
                     i,EnumToString(CheckPointer(shapes[i])));
        }
     }

//--- Oluşturulmuş tüm dinamik nesneleri silmeliyiz
   for(int i=0;i<total;i++)
     {
      //--- Sadece POINTER_DYNAMIC tipli nesneleri silebiliriz
      if(CheckPointer(shapes[i])==POINTER_DYNAMIC)
        {
         //--- Silindiğinde uyar
         PrintFormat("shapes[%d] siliniyor",i);
         //--- Bir nesneyi işaretçisi aracılığı ile sil
         delete shapes[i];
        }
     }
  }
```

Bir nesne delete operatörü ile silindiğinde, işaretçi tipinin kontrol edilmesi gerektiğini lütfen not edin. Sadece POINTER_DYNAMIC tipine sahip işaretçiler "delete" kullanılarak silinebilir. Diğer işaretçi tiplerinde ise bir hata dönülecektir.

Ama polimorfizm, fonksiyonların kalıtım sırasında yeniden tanımlanmasının yanında, bir sınıf içindeki farklı parametre kümelerine sahip olan aynı fonksiyonların uygulamasını da içerir. Bu, sınıfın aynı isimde ama farklı tip ve/veya parametre kümesine sahip fonksiyonlara sahip olabileceği anlamına gelir. Bu durumda polimorfizm fonksiyonların aşırı-yüklenmesi yoluyla uygulanır.

Ayrıca Bakınız

Standart Kütüphane

Aşırı Yükleme

Aynı ismi kullanan ama parametre sayıları farklı olan iki veya daha fazla yöntemin tek bir sınıf içinde tanımlanması mümkündür. Buna yöntemlerin aşırı-yüklenmesi denir ve ilgili yöntemlerin aşırı yüklenmiş olduğu söylenir.

Yöntemlerin aşırı-yüklenmesi polimorfizmin şekillerinden biridir. Fonksiyonların aşırı-yüklenmesi ile aynı kurallara göre gerçekleştirilir.

Eğer çağrılan fonksiyon hiçbir eşleşmeye sahip değilse, derleyici uygun bir fonksiyon için ardışık üç aşama ile arama yapar:

1. sınıf yöntemleri içinde arama;
2. temel sınıf yöntemleri içinde arama (en yakın ebeveynden ilk baştakine kadar durmadan);
3. diğer fonksiyonlar arasında arama.

Tam bir karşılık bulunamamışsa ama farklı aşamalarda birkaç uygun fonksiyon bulunmuşsa, en düşük seviyede bulunan fonksiyon kullanılır. Tek bir seviyede birden fazla uygun fonksiyon olamaz.

Ayrıca Bakınız

Fonksiyonların Aşırı Yüklenmesi

Sanal Fonksiyonlar

'virtual' (sanal) anahtar kleimesi bir fonksiyon belirtecidir. Temel ve türetik sınıfların fonksiyonları arasından uygun bir fonksiyon üyesinin, çalışma sırasında dinamik olarak seçilmesini sağlayan bir mekanizma sunar. Yapılar sanal fonksiyonlara sahip olamazlar. Bu sadece fonksiyon üyeleri için yapılacak bildirimleri değiştirmek için kullanılır.

Sanal fonksiyonlar tıpkı sıradan fonksiyonlar gibi bir çalıştırılabilir gövdeye sahip olmalıdır. Çağrıldığında anlamsallığı diğer fonksiyonlarla aynıdır.

Sanal fonksiyonlar türetik sınıflarda geçersiz olabilir. Sanal fonksiyon için hangi fonksiyon tanımının çağrılması gerektiğine dinamik olarak (çalışma sırasında) karar verilir. Temel sınıfın bir sanal fonksiyon içermesi ve türetik sınıfların da bu fonksiyonun kendilerine özgü versiyonlarını içermeleri tipik bir durumdur.

Temel sınıf işaretçisi bir temel sınıf nesnesini veya bir türetik sınıf nesnesini gösterebilir. Çağrılacak üye fonksiyonun seçimi çalışma sırasında gerçekleştirilir. Bu seçim işaretçi tipine değil nesne tipine bağlıdır. Bir türetik sınıfın hiç üyesi yoksa, varsayılan olarak temel sınıfın sanal fonksiyonu kullanılır.

Yıkıcılar, virtual anahtar sözcüğü ile bildirilmiş olduklarına bakılmaksızın, daima sanaldır.

MT5_Tetris.mq5 örneğindeki kullanımı inceleyelim. Temel sınıf CTetrisShape, 'Draw' sanal fonksiyonuyla birlikte, dahil edilmiş MT5_TetrisShape.mqh dosyasında tanımlanmıştır.

```
//+------------------------------------------------------------------+
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[]);
  };
```

Daha sonra her bir türetik sınıf için bu fonksiyon türetik sınıfın niteliklerine göre uygulanır. Örneğin ilk şekil CTetrisShape1, Draw() fonksiyonunun kendine has uygulamasına sahiptir:

```
class CTetrisShape1 : public CTetrisShape
  {
public:
   //--- şekil çizimi
   virtual void      Draw()
     {
      int    i;
      string name;
      //---
      if(m_turn==0 || m_turn==2)
        {
         //--- yatay
         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
        {
         //--- dikey
         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);
           }
        }
     }
  }
```

Kare şekli, CTetrisShape6 aracılığı ile tarif edilmiştir ve Draw() yönteminin kendine has bir uygulamasına sahiptir:

```
class CTetrisShape6 : public CTetrisShape
  {
public:
   //--- Şekil çizimi
   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);
        }
     }
  };
```

Oluşturulan nesneyi içeren sınıfa bağlı olarak, şu veya bu türetik sınıfın sanal fonksiyonu çağrılır.

```
void CTetrisField::NewShape()
  {
//--- 7 olası şekilden rassal olarak birinin oluşturulması
   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;
     }
//--- çiz
   m_shape.Draw();
//---
  }
```

'override' Şekillendiricisi

'override' şekillendiricisi ile bildirilen fonksiyonlar, ebeveyn sınıf yönteminin etkisiz kılınmasını sağlar. Bu tekniğin kullanılması, ilgili hataların önlenmesini sağlar. Örneğin, yöntem işaretinin yanlışlıkla değiştirilmesini önleyebilir. 'func' yönteminin temel sınıf bünyesinde kullanıldığını düşünelim ve yöntem, argüman olarak int tipli bir değişken içersin:

```
class CFoo
  {
   void virtual func(int x) const { }
  };
```

Sonra yöntem çocuk sınıf içinde geçersiz kılınsın:

```
class CBar : public CFoo
  {
   void func(short x) { }
  };
```

Görüldüğü gibi, argümandaki 'int' tipi 'short' tipiyle yanlışlıkla değiştirildi. Aslında burada geçersiz kılma işlemi değil, aşırı yükleme işlemi yapıldı. Yani, aşırı yüklenmiş fonksiyon tanımlama algoritmasına göre hareket edildiğinde, belli durumlarda derleyici temel sınıf yöntemini aşırı yüklenmiş yönteme tercih edecektir.

Bu tip hataları önlemek için, ilgili yönteme açık şekilde 'override' şekillendiricisini eklemeniz gerekir.

```
class CBar : public CFoo
  {
   void func(short x) override { }
  };
```

Yöntemin işareti geçersiz-kılma sırasında değiştirilirse, derleyici ebeveyn sınıfta aynı işarete sahip bir yöntem bulamaz ve hata verir:

```
'CBar::func' method is declared with 'override' specifier but does not override any ba
```

'final' Şekillendiricisi

'final' şekillendiricisi tamamen aksi yönde çalışır ve yöntemin çocuk sınıf içinde geçersiz kılınmasını engeller. Bir yöntemin tamamen yeterli olduğundan eminseniz, daha sonradan değiştirilmemesi için yöntemi 'final' şekillendiricisi ile bildirin.

```
class CFoo
  {
   void virtual func(int x) final { }
  };

class CBar : public CFoo
  {
   void func(int) { }
  };
```

Yöntemi yukarıdaki örnekte gösterildiği gibi 'final' şekillendiricisi ile etkisiz kılmak istediğinizde derleyici hata dönüşü yapar:

```
'CFoo::func' method declared as 'final' cannot be overridden by 'CBar::func'
see declaration of 'CFoo::func'
```

Ayrıca bakınız

Standart Kütüphane

Bir Sınıfın/Yapının statik üyeleri

Statik üyeler

Sınıf üyeleri, static bellek sınıfı şekillendiricisi kullanılarak bildirilebilir. Bu veri üyeleri sınıfın tüm örnekleri tarafından paylaşılır ve bir yerde depolanırlar. Statik olmayan veri üyeleri her bir sınıf nesnesi değişkeni için oluşturulurlar.

Statik üyelerin bildirilememesi, ilgili değişkenlerin global kapsamda bildirilmesini gerektirebilir. Bu, veri ile sınıf arasındaki ilişkiyi koparır ve NYP'nin temel paradigmasıyla (değişkenleri ve yöntemleri bir sınıfta bir araya getirme) uyumlu değildir. Statik üye, belirli bir örneğe özgü olmayan sınıf verisinin, sınıf çerçevesinde var olmasına izin verir.

Statik üye belirli örneğe bağlı olmadığından referansı şu şekilde olur:

```
sınıf_ismi::değişkenler
```

Burada *sınıf_ismi* sınıfın, *değişken* ise sınıf üyesinin ismidir.

Gördüğünüz gibi, sınıfların statik üyelerine erişmek için kavram çözümleme operatörü :: kullanılır. Sınıf yöntemleri içinde bir statik üyeye erişim gerçekleştirdiğinizde, kavram operatörünün kullanımı isteğe bağlıdır.

Statik sınıf üyesinin istenen değerler ile açık şekilde başlatılması gerekir. Yani, üyenin bildirimi ve başlatımı global kapsamda yapılmalıdır. Statik üyelerin başlatma sırası, global kapsamdaki bildirimlerinin sırasına karşılık gelir.

Metin ayrıştırma için kullanılacak bir sınıfımız (*CParser*) olduğunu düşünelim ve işlenen kelime ve karakter sayısını hesaplamamız gereksin. Tek yapmamız gereken sınıf üyelerini statik olarak tanımlamak ve bunları global düzeyde başlatmaktır. Sonrasında sınıfın tüm örnekleri, kelime ve karakterlerin ortak sayaçlarını kullanacaktır.

```
//+------------------------------------------------------------------+
//| Sınıf "Metin Analizci"                                           |
//+------------------------------------------------------------------+
class CParser
  {
public:
   static int        s_words;
   static int        s_symbols;
   //--- Yapıcı ve yıkıcı
                     CParser(void);
                    ~CParser(void){};
  };
...
//--- Parser (ayrıştırıcı) sınıfının statik üyelerinin global seviyede başlatılması
int CParser::s_words=0;
int CParser::s_symbols=0;
```

Bir statik sınıf üyesi *const* anahtar kelimesi ile bildirilebilir. Bunun gibi statik sabitler, global düzeyde ve '*const*' anahtar kelimesiyle başlatılır:

```
//+------------------------------------------------------------------+
//| İşlenmiş veriyi depolamak için "Yığın" sınıfı                    |
//+------------------------------------------------------------------+
class CStack
  {
public:
                     CStack(void);
                    ~CStack(void){};
...
private:
   static const int  s_max_length; // Maksimum yığın kapasitesi
  };

//--- CStack sınıfının statik sabitinin başlatılması
const int CStack::s_max_length=1000;
```

this İşaretçişi

Anahtar sözcük this üstü kapalı şekilde bildirilmiş, kendisine (yöntemin uygulandığı bağlamda belirli bir sınıf örneğine) yönelik bir işaretçiyi belirtir. Sınıfın sadece statik olmayan yöntemlerinde kullanılabilir. 'this' işaretçisi gizli ve statik olmayan bir sınıf üyesidir.

Statik fonksiyonlarda sınıfın sadece statik üyelerine/yöntemlerine erişebilirsiniz.

Statik yöntemler

MQL5 dilinde static tipli üye fonksiyonları kullanılabilir. '*static*' şekillendiricisi fonksiyonun sınıf içindeki bildiriminde dönüş tipinden önce yer almalıdır.

```
class CStack
  {
public:
   //--- Yapıcı ve yıkıcı
                     CStack(void){};
                    ~CStack(void){};
   //--- Maksimum yığın kapasitesi
   static int        Capacity();
private:
   int               m_length;     // Yığındaki elemanların sayısı
   static const int  s_max_length; // Maksimum yığın kapasitesi
  };
//+------------------------------------------------------------------+
//| Yığında depolanan elemanların sayısına dönüş yapar               |
//+------------------------------------------------------------------+
int CStack::Capacity(void)
  {
   return(s_max_length);
  }
//--- CStack sınıfının statik sabitinin başlatılması
const int CStack::s_max_length=1000;
```

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- CStack tipli değişkenin bildirimini yap
   CStack stack;
//--- nesnenin statik yöntemini çağır
   Print("CStack.s_max_length=",stack.Capacity());
//--- yöntem statik olduğu ve nesnenin varlığına ihtiyaç duymadığı için şu şekilde de
   Print("CStack.s_max_length=",CStack::Capacity());
  }
```

const şekillendiricisi ile bildirilen yöntemler, sabit olarak isimlendirilir ve sınıftaki gizli üyeleri şekillendiremezler. Sınıfın sabit fonksiyonları ve sabit parametrelerinin bildirimi *const-correctness* (sabit doğruluk) kontrolü olarak adlandırılır. Bu kontrol sayesinde derleyicinin nesne değerlerinin kararlılığını temin edeceğine ve yanlış bir şey varsa, derleme sırasında bir hata dönüşü yapacağına emin olabilirsiniz.

const şekillendiricisi, sınıf bildirimlerinde argüman listesinden sonra gelir. Sınıf dışındaki tanımlar da, aynı şekilde *const* şekillendiricisini içermelidir:

```
//+------------------------------------------------------------------+
//| Sınıf "Dikdörtgen"                                               |
//+------------------------------------------------------------------+
class CRectangle
  {
private:
   double            m_height;      // Yükseklik
   double            m_height;      // Yükseklik
public:
   //--- Yapıcılar ve yıkıcı
                     CRectangle(void):m_width(0),m_height(0){};
                     CRectangle(const double w,const double h):m_width(w),m_height(h)
                    ~CRectangle(void){};
   //--- Alanın hesaplanması
   double            Square(void) const;
   static double     Square(const double w,const double h);// { return(w*h); }
  };
//+------------------------------------------------------------------+
//| "Dikdörtgen" nesnesinin alanına dönüş yapar                      |
//+------------------------------------------------------------------+
double CRectangle::Square(void) const
  {
   return(Square(m_width,m_height));
  }
//+------------------------------------------------------------------+
//| İki değişkenin çarpımına dönüş yapar                             |
//+------------------------------------------------------------------+
static double CRectangle::Square(const double w,const double h)
```

```
  {
   return(w*h);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Kenarları 5 ve 6'ya eşit bir rect dikdörtgen oluştur
   CRectangle rect(5,6);
//--- bir sabit yöntem kullanarak dikdörtgen alanını bul
   PrintFormat("rect.Square()=%.2f",rect.Square());
//--- CRectangle sınıfının statik yöntemini kullanarak sayıların çarpımlarını bul
   PrintFormat("CRectangle::Square(2.0,1.5)=%f",CRectangle::Square(2.0,1.5));
  }
```

Sabitlik kontrolü kullanmanın bir diğer faydası da, derleyicinin bu durumda özel bir optimizasyon oluşturmasıdır. Örneğin, sabit nesneyi salt okunabilir hafızaya yerleştirir.

Statik fonksiyonlar const şekillendiricisi ile belirlenemez; çünkü bu şekillendirici, fonksiyon çağrıldığı zaman örnek üyelerinin sabitliğini garanti eder. Ama yukarıda da belirtildiği gibi statik fonksiyon statik olmayan sınıf üyelerine erişemez.

Ayrıca Bakınız

Statik Değişkenler, Değişkenler, Referanslar: Şekillendirici & ve Anahtar Sözcük this

Fonksiyon Şablonları

Aşırı-yüklenmiş fonksiyonlar genellikle benzer işlemleri gerçekleştirmek için kullanılır. ArraySize(), MQL5 içinde böyle bir fonksiyona örnektir. Herhangi bir dizinin büyüklüğüne dönüş yapar. Gerçekte bu sistem fonksiyonu aşırı yüklenmiştir ve bu aşırı-yükleme uygulamasının tamamı MQL5 yazılım geliştiricilerinden saklanmıştır:

```
int  ArraySize(
   void&  array[]      // kontrol edilen dizi
   );
```

Yani MQL5 derleyicisi, fonksiyon çağrıldığında gerekli olanı yapacaktır. Örnek olarak, bunun tamsayı tipli diziler için nasıl yapılabileceğini gösterelim:

```
int  ArraySize(
   int&  array[]      // int tipi elemanların dizisi
   );
```

ArraySize() fonksiyonu, tarihsel veri şeklindeki kotasyonlarla çalışabilmek amacıyla, MqlRates tipi bir dizi ile şu şekilde gösterilebilir :

```
int  ArraySize(
   MqlRates&  array[] // MqlRates tipi verilerle doldurulmuş dizi
   );
```

Bu şekilde, farklı tipteki verilerle çalışmak için aynı fonksiyonu kullanmak çok uygun bir seçenek haline gelir. Ama bunun için tüm ön çalışma yapılmış olmalıdır. İlgili fonksiyon, doğru şekilde çalışması gereken tüm veri tipleri için aşırı-yüklenmiş olmalıdır.

Uygun bir çözüm vardır. Benzer işlemlerin tüm veri tipleri için çalıştırılması gerekiyorsa fonksiyon şablonları kullanılabilir. Programcı sadece bir fonksiyon şablonu tarifi yazmalıdır. Şablonu tarif ederken fonksiyonun birlikte çalışması gereken kesin veri tipleri yerine, sadece bazı biçimsel parametreleri tanımlamalıyız. Derleyici, tüm tiplerin doğru işlenmesi için, fonksiyon çağrısında kullanılan argümanların tiplerini temel alarak, otomatik olarak çeşitli fonksiyonlar oluşturur.

Fonksiyon şablonu tanımı template anahtar kelimesi ile başlar ve açı parantezler arasındaki biçimsel parametrelerin listesi ile devam eder. Biçimsel parametreler typename anahtar kelimesinden sonra gelir . Biçimsel parametreler sistemde gömülü veya kullanıcı tanımlı tiplerdir. Şu amaçlarla kullanılırlar:

- fonksiyon argümanlarının tiplerini belirtmek,
- fonksiyonun dönüş tipini belirtmek,
- değişkenleri fonksiyon tanımında bildirmek

Şablon parametrelerinin sayısı sekizi geçemez. Şablon tanımındaki biçimsel parametreler en az bir kere fonksiyon parametreleri içinde yer almalıdır. Biçimsel parametrelerin isimleri benzersiz olmalıdır.

Aşağıda, sayısal tipli (tamsayı ve reel sayılar) bir dizide en büyük değeri aramak için bir fonksiyon şablonu örneği yer almaktadır:

```
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);
  }
```

Bu şablon, geçirilmiş dizideki en yüksek değeri bulan ve sonuç olarak bu değere dönüş yapan fonksiyonu tanımlar. Lütfen not edin: MQL5 içindeki ArrayMaximum() fonksiyonu, en yüksek değere değil, bu değeri bulmakta kullanılabilecek indise dönüş yapacaktır. Örnek olarak:

```
//--- bir dizi oluştur
   double array[];
   int size=50;
   ArrayResize(array,size);
//---  rassal değerler ile doldur
   for(int i=0;i<size;i++)
     {
      array[i]=MathRand();
     }

//--- dizideki en yüksek değerin pozisyonunu bul
   int max_position=ArrayMaximum(array);
//--- şimdi en yüksek değerin kendisini al
   double max=array[max_position];
//--- bulunan değeri göster
   Print("Max value = ",max);
```

Böylece, dizideki değeri bulmak için iki aşamalı bir yol izledik. ArrayMax() fonksiyon şablonuyla, sadece uygun tipte bir diziyi fonksiyona geçirerek, gerekli tipteki sonucu alabiliriz. Yani, son iki satırın yerine

```
//--- dizideki en yüksek değerin pozisyonunu bul
   int max_position=ArrayMaximum(array);
//--- şimdi dizideki en yüksek değerin kendisini al
   double max=array[max_position];
```

Artık, dönüş tipinin, fonksiyona geçirilen dizinin tipiyle aynı olduğu tek bir satır kullanabiliriz:

```
//--- en yüksek değeri bul
   double max=ArrayMax(array);
```

Bu durumda, ArrayMax() fonksiyonunun dönüş tipi otomatik olarak dizi tipiyle eşleşecektir.

Çeşitli veri tipleriyle genel amaçlı çalışma yöntemleri oluşturmak ve argüman tipini 'string' olarak almak için typename anahtar sözcüğünü kullanın. Veri tipini dizgi olarak döndüren özel bir fonksiyon örneğini göz önüne getirelim:

```
#include <Trade\Trade.mqh>
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   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));
//---
  }
//+------------------------------------------------------------------+
//| Tip, bir satır şeklinde döndürülür                               |
//+------------------------------------------------------------------+
template<typename T>
string GetTypeName(const T &t)
  {
//--- tipi bir satır şeklinde döndür
   return(typename(T));
//---
  }
```

Fonksiyon şablonları sınıf yöntemleri için de kullanılabilirler, örneğin:

```
class CFile
  {
  ...
public:
  ...
   template<typename T>
   uint WriteStruct(T &data);
  };

template<typename T>
uint CFile::WriteStruct(T &data)
  {
  ...
   return(FileWriteStruct(m_handle,data));
  }
```

Fonksiyon şablonları, export, virtual ve #import anahtar sözcükleri ile bildirilmemelidir.

Şablon fonksiyonunun aşırı yüklenmesi

Bazen şablon fonksiyonların da aşırı yüklenmesi gerekebilir. Örneğin tip dönüştürme kullanarak ikinci parametrenin değerini birinciye atayan bir şablon fonksiyonumuz olabilir. MQL5, string ve bool tipleri arasında tip dönüşümüne izin vermez.. Bunu kendimiz yapabiliriz - şimdi bir şablon fonksiyonunu aşırı yükleyelim. Örneğin:

```
//+------------------------------------------------------------------+
//| Şablon fonksiyon                                                 |
//+------------------------------------------------------------------+
template<typename T1,typename T2>
string Assign(T1 &var1,T2 var2)
  {
   var1=(T1)var2;
   return(__FUNCSIG__);
  }
//+------------------------------------------------------------------+
//| bool+string için özel aşırı yükleme                              |
//+------------------------------------------------------------------+
string Assign(bool &var1,string var2)
  {
   var1=(StringCompare(var2,"true",false) || StringToInteger(var2)!=0);
   return(__FUNCSIG__);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int i;
   bool b;
   Print(Assign(i,"test"));
   Print(Assign(b,"test"));
  }
```

Kod çalıştırıldığında, sonuç olarak Assign() şablonunun int+string çifti için kullanıldığını görebiliriz, aşırı yüklenmiş versiyon ise ikinci çağrı sırasında bool+string çifti için zaten kullanmış durumda.

```
string Assign<int,string>(int&,string)
string Assign(bool&,string)
```

Ayrıca bakınız

Aşırı Yükleme

Şablon avantajları

Fonksiyon şablonları çeşitli veri tiplerinde benzer işlemler yapmak istediğinizde kullanılır (bir dizideki maksimum değerli elmanı bulmak gibi). Şablonların avantajı, her farklı tip için ayrı aşırı yükleme yapmanızı gerektirmemesidir. Her farklı veri tipi için aşırı yükleme yapmak yerine

```
double ArrayMax(double array[])
  {
  ...
  }
int ArrayMax(int array[])
  {
  ...
  }
uint ArrayMax(uint array[])
  {
  ...
  }
long ArrayMax(long array[])
  {
  ...
  }
datetime ArrayMax(datetime array[])
  {
  ...
  }
```

sadece bir şablon fonsiyonu yazmamız yeterli

```
template<typename T>
T ArrayMax(T array[])
  {
   if(ArraySize()==0)
      return(0);
   uint max_index=ArrayMaximum(array);
   return(array[max_index]);
  }
```

bunu kaynak kodunuzda kullanabilmeniz için:

```
double high[];
datetime time[];
....
double max_high=ArrayMax(high);
datetime lasttime=ArrayMax(time);
```

Burada, kullanılan verinin tipini belirten *T* biçimsel parametresi, derleme esnasında gerçekte uygulanan tip ile değiştirilir. Yani derleyici herbir tip için (double, datetime, vb) otomatik olarak bir fonksiyon oluşturur. MQL5 dili, yaklaşımın tüm avantajları ile sınıf sablonları oluşturmanıza olanak sağlar.

Sınıf şablonları

Sınıf şablonları template anahtar sözcüğü ve açı parantezlerle <>, typename anahtar sözcüğü ile biçimsel parametreler sıralanarak bildirilir. Bu girdi, bir sınıf uygulaması yaparken, bir reel değişkeni tanımlayan *T* biçimsel parametresi ile bir jenerik sınıf üzerinde çalışıldığı konusunda derleyiciyi bilgilendirir. Örnek olarak *T* tipli elemanlardan oluşan bir diziyi depolamak için bir vektör sınıfı oluşturalım:

```
#define TOSTR(x) #x+" "   // bir nesne ismini göstermek için makro
//+------------------------------------------------------------------+
//| T-tipi bileşenleri depolamak için bir vektör sınıfı              |
//+------------------------------------------------------------------+
template <typename T>
class TArray
  {
protected:
   T                 m_array[];
public:
   //--- yapıcı varsayılan olarak 10 elemanlı bir dizi oluşturur
   void TArray(void){ArrayResize(m_array,10);}
   //--- belirtilen dizi boyutuna göre dizi oluşturmak için bir yapıcı
   void TArray(int size){ArrayResize(m_array,size);}
   //--- TArray tipli nesnede depolanan verinin tipine ve miktarına dönüş yap
   string Type(void){return(typename(m_array[0])+":"+(string)ArraySize(m_array));};
  };
```

Şimdi çeşitli veri tipleriyle çalışmak için farklı yöntemler kullanarak üç *TArray* nesnesi oluşturalım

```
void OnStart()
  {
   TArray<double> double_array;   // vektörün varsayılan boyutu 10
   TArray<int> int_array(15);     // vectör boyutu 15
   TArray<string> *string_array;  // TArray<string> vector için işaretçi
//--- dinamik nesne oluştur
   string_array=new TArray<string>(20);
//--- nesne ismini veri tipini ve boyutunu günlükte görüntüle
   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());
//--- program tamamlanmadan bir dinamik nesneyi sil
   delete(string_array);
  }
```

Çalıştırılan betiğin sonuçları:

```
  double_array  (double:10)
  int_array  (int:15)
  string_array  (string:20)
```

Artık, farklı veri tiplerine sahip (double, int ve string) üç vektörümüz var.

Sınıf şablonları taşıyıcıların (farklı tiplerdeki nesneleri kapsüllemek için kullanılan nesneler) geliştirilmesi için uygundur. Taşıyıcı nesneler belli tipteki diğer nesneleri içerirler. Genellikle depolanan verilerle yapılan çalışmalar hemen taşıyıcıya eklenir.

Örneğin, dizi dışındaki elemanlara erişimi kısıtlayan bir sınıf şablonu oluşturabilir ve "kapsam dışı" kritik halarını önleyebilirsiniz.

```
//+------------------------------------------------------------------+
//| Bir dizi elamnına serbest erişim sağlayan bir sınıf              |
//+------------------------------------------------------------------+
template<typename T>
class TSafeArray
  {
protected:
   T                 m_array[];
public:
   //--- varsayılan yapıcı
   void              TSafeArray(void){}
   //--- belirtilen boyutta dizi oluştumak için yapıcı
   void              TSafeArray(int size){ArrayResize(m_array,size);}
   //--- dizi boyutu
   int               Size(void){return(ArraySize(m_array));}
   //--- dizi boyutunu değiştir
   int               Resize(int size,int reserve){return(ArrayResize(m_array,size,rese
   //--- diziyi boşalt
   void              Erase(void){ZeroMemory(m_array);}
   //--- indise göre dizi elemanına erişim sağlayan operatör
   T                 operator[](int index);
   //--- dizinin tüm elemanlarını tekseferde alabilmek için atama operatörü
   void              operator=(const T  &array[]); // T tipli dizi
  };
//+------------------------------------------------------------------+
//| İndisine göre eleman alma                                        |
//+------------------------------------------------------------------+
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, %d indisi (0-%d) kapsamında değil!",__FUNCTION__,index,max);
      return(invalid_value);
     }
//---
   return(m_array[index]);
  }
//+------------------------------------------------------------------+
```

```
//| Dizi için atama                                                  |
//+------------------------------------------------------------------+
template<typename T>
void TSafeArray::operator=(const T  &array[])
  {
   int size=ArraySize(array);
   ArrayResize(m_array,size);
//--- T tipi, kopyalama işlemini desteklemeli
   for(int i=0;i<size;i++)
      m_array[i]=array[i];
//---
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int copied,size=15;
   MqlRates rates[];
//--- fiyat dizisini kopyala
   if((copied=CopyRates(_Symbol,_Period,0,size,rates))!=size)
     {
      PrintFormat("CopyRates(%s,%s,0,%d) çağrısı %d hata koduna dönüş yaptı",
      _Symbol,EnumToString(_Period),size,GetLastError());
      return;
     }
//--- bir taşıyıcı oluşturup MqlRates dizisini ekle
   TSafeArray<MqlRates> safe_rates;
   safe_rates=rates;
   //--- dizi kapsamında bir indis
   int index=3;
   PrintFormat("Close[%d]=%G",index,safe_rates[index].close);
   //--- indis dizi kapsamı dışında
   index=size;
   PrintFormat("Close[%d]=%G",index,safe_rates[index].close);
  }
```

Şablon bildirimlerinin sınıf bildirimleri dışında yöntem tanımlarken de kullanılabileceğini not ediniz:

```
template<typename T>
T TSafeArray::operator[](int index)
  {
  ...
  }
template<typename T>
void TSafeArray::operator=(const T  &array[])
  {
  ...
  }
```

Sınıf ve fonksiyon şablonları, virgülle eyrılmış şekilde biçimsel parametreler kullanabilmenizi sağlar. Örneğin, "anahtar - değer" çiftlerini kaydetmek için bağıntılar:

```
template<typename Key, template Value>
class TMap
  {
  ...
  }
```

Ayrıca bakınız

Fonksiyon şablonları, Aşırı Yükleme

Soyut Sınıflar ve Saf Sanal Fonksiyonlar

Soyut sınıflar jenerik yapılar oluşturmak için kullanılır. Bu şekilde daha çok özelleştirilmiş türetik sınıfların yazılabilmesi sağlanır. Soyut sınıflar sadece başka sınıflar için temel sınıf olarak kullanılabilir. Bu nedenle soyut sınıf tipini kullanarak nesne oluşturmak olanaksızdır.

İçeriğinde en az bir saf sanal fonksiyon barındıran sınıflara soyut sınıf denir. Soyut sınıflardan türetilen tüm sınıflar temel sınıfın tüm saf sanal fonksiyonlarını kullanmalıdır, aksi durumda bu sınıflar da soyut sınıf olurlar.

. CAnimal sınıfını örnek verelim. Bu sınıf sadece bazı genel fonksiyonları sağlar - CAnimal tipli nesneler pratik kullanım için fazla geneldir. Bu da CAnimal'ı soyut sınıflar için iyi bir örnek yapar:

```
class CAnimal
  {
public:
                        CAnimal();      // Yapıcı
   virtual void         Sound() = 0;    // Saf sanal fonksiyon
private:
   double               m_legs_count;   // Hayvanın bacak sayısı
  };
```

Burada Sound() fonksiyonu bir saf sanal fonksiyondur çünkü saf sanal fonksiyon belirteci PURE (=0) ile bildirilmiştir.

Saf sanal fonksiyonlar sadece PURE belirteci ayarlandığı için sanal fonksiyondur: (=NULL) veya (=0). Soyut sınıf bildirimi ve kullanımı için bir örnek:

```
class CAnimal
  {
public:
   virtual void         Sound()=NULL;   // PURE (saf) yöntem, türetik sınıf içinde iptal
  };
//--- Soyut sınıftan türetilmiş
class CCat : public CAnimal
 {
public:
  virtual void          Sound() { Print("Myau"); } // PURE yöntem iptal edildi, CCat art
 };

//--- Hatalı kullanım örnekleri
new CAnimal;          // 'CAnimal' hatası - derleyici "soyut sınıf somutlaştırılamaz" h
CAnimal some_animal; // 'CAnimal' hatası - derleyici "soyut sınıf somutlaştırılamaz" h

//--- Doğru kullanım örneği
new CCat;  // Hata yok - CCat sınıfı soyut değil
CCat cat;  // Hata yok - CCat sınıfı soyut değil
```

Soyut sınıflar üzerindeki kısıtlamalar

Bir soyut sınıfın yapıcı fonksiyonunun bir saf sanal fonksiyonu çağırması durumunda (doğrudan veya dolaylı olarak) sonuç tanımsızdır.

```
//+------------------------------------------------------------------+
//| Soyut baz sınıf                                                  |
//+------------------------------------------------------------------+
class CAnimal
  {
public:
   //--- Saf sanal fonksiyon
   virtual void      Sound(void)=NULL;
   //--- Fonksiyon
   void              CallSound(void) { Sound(); }
   //--- Yapıcı
   CAnimal()
    {
     //--- Sanal yöntem çağrısının açık şekli
     Sound();
     //--- Sanal yöntem çağrısının gizli şekli (üçüncü bir fonksiyon ile)
     CallSound();
     //--- Yapıcılar ve yıkıcılar türetik sınıf içinden çağrılan bir fonksiyon ile ipt
     //--- veya sanal olsalar bile daima kendi fonksiyonlarını çağırırlar
     //--- çağrılan fonksiyon saf sanal ise,
     //--- çağrı kritik hataya sebep olacaktır: "saf sanal fonksiyon çağrısı"
    }
  };
```

Ama soyut sınıflardaki yapıcı ve yıkıcılar diğer fonksiyonları çağırabilirler.

Ad Alanları

Ad alanı, içinde çeşitli kimliklerin tanımlandığı özel olarak bildirilen bir alandır: değişkenler, fonksiyonlar, sınıflar, vb. namespace anahtar sözcüğü kullanılarak ayarlanır:

```
namespace name of_space {
  // fonksiyon, sınıf ve değişken tanımları listesi
}
```

'namespace' uygulamak, global ad alanını alt alanlara bölmeyi sağlar. Ad alanı içindeki tüm kimlikler ek bilgi belirtilmeden birbirleri için kullanılabilir. :: operatörü (bağlam çözünürlüğü işlemi), ad alanı üyelerine dışarıdan erişmek için kullanılır.

```
namespace ProjectData
{
class DataManager
  {
public:
   void              LoadData() {}
  };
void Func(DataManager& manager) {}
}
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ProjectData ad alanıyla çalışma
   ProjectData::DataManager mgr;
   mgr.LoadData();
   ProjectData::Func(mgr);
  }
```

Ad alanları, mantıksal gruplar biçiminde bir kod düzenlemek ve bir programda birkaç kütüphane kullanıldığında ortaya çıkabilecek ad çakışmalarını önlemek için kullanılır. Bu gibi durumlarda, her bir kütüphanenin gerekli fonksiyonlarına ve sınıflarına açıkça erişmek için her bir kütüphane kendi ad alanında bildirilebilir.

Bir ad alanı, bir veya birkaç dosyada birkaç blok halinde bildirilebilir. Derleyici ön işleme sırasında tüm parçaları bir araya getirir ve elde edilen ad alanı tüm parçalarda bildirilen bütün üyeleri içerir. Sample.mqh include (ekle) dosyasında uygulanan bir A sınıfımız olduğunu varsayalım:

```
//+------------------------------------------------------------------+
//|                                                   Sample.mqh |
//+------------------------------------------------------------------+
class A
  {
public:
                     A() {Print(__FUNCTION__);}
  };
```

Bu sınıfı projemizde kullanmak istiyoruz ama zaten A sınıfımız var. Her iki sınıfı da kullanabilmek ve kimlik çakışmasından kaçınmak için, eklenen dosyayı bir ad alanına kaydırın:

```
//--- ilk A sınıfını bildir
class A
  {
public:
                     A() {Print(__FUNCTION__);}
  };
//--- çakışmayı önlemek için A sınıfını Sample.mqh dosyasından Library ad alanına kayd
namespace Library
{
#include "Sample.mqh"
}
//--- Library ad alanına başka bir sınıf daha ekle
namespace Library
{
class B
  {
public:
                     B() {Print(__FUNCTION__);}
  };
}
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- global ad alanından A sınıfını kullan
   A a1;
//--- Library ad alanından A ve B sınıflarını kullan
   Library::A a2;
   Library::B b;
  }
//+------------------------------------------------------------------+

/*
Result:
   A::A
   Library::A::A
   Library::B::B
*/
```

Ad alanları iç içe yerleşebilir. İçi içe yerleşmiş bir ad alanı, üst alan üyelerine sınırsız erişime sahiptir; ancak üst alan üyeleri iç içe yerleşmiş ad alanına sınırsız erişime sahip değildir.

```
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;}
}
```

Global ad alanı

Kimlik, ad alanında açıkça belirtilmezse, global ad alanının örtülü bir parçası olarak kabul edilir. Global kimliği açıkça belirlemek için, bir ad olmadan kapsam çözünürlüğü operatörünü kullanın. Bu durum, bu kimliği farklı bir ad alanında bulunan aynı ada sahip diğer herhangi bir öğeden ayırt etmenizi sağlayacaktır. Örneğin, bir fonksiyonu içe aktarırken:

```
#import "lib.dll"
int Func();
#import
//+------------------------------------------------------------------+
//|  Bazı fonksiyonlar                                               |
//+------------------------------------------------------------------+
int Func()
  {
   return(0);
  }
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//+--- içe aktarılan fonksiyonu çağır
   Print(lib::Func());
//+--- fonksiyonumuzu çağır
   Print(::Func());
  }
```

Bu durumda, DLL işlevinden içe aktarılan tüm fonksiyonlar aynı adın ad alanına eklenmiştir. Bu, derleyicinin çağrılacak fonksiyonu net bir şekilde belirlemesini sağladı.

Ayrıca bakınız

Global Değişkenler, Yerel Değişkenler, Değişkenlerin kapsamı ve ömrü, Objelerin oluşturulması ve silinmesi

Standart Sabitler, Sayımlar ve Yapılar

Program yazımını basitleştirmek için ve program metinlerini daha anlaşılır kılmak için, MQL5 dili ön tanımlı standart sabitler ve sayımlar sunmaktadır. Ayrıca, servis yapıları da bilgi saklamak için kullanılır.

Standart sabitler makrolara benzer ve int tipindedirler.

Sabitler amaçlarına göre gruplanır:

- Çizelge sabileri fiyat çizelgeleri ile çalışılırken kullanılır: açma, konumlandırma, parametrelerin ayarlanması gibi;
- Nesne sabitleri çizelgede oluşturulup görüntülenebilen grafiksel nesnelerin işlenebilmesi için düşünülmüştür;
- Gösterge sabitleri standart ve özel göstergelerle çalışmak için kullanılır;
- Çevre durumu sabitleri bir MQL5-programının özelliklerini tarif eder, müşteri terminali, finansal enstrümanlar ve mevcut hesapla ilgili bilgi gösterir;
- Alım-satım sabitleri, alım-satım sırasında çeşitli bilgilerin belirltilmesine olanak tanır;
- İsimlendirilmiş sabitler MQL5 dilinin sabitleridir;
- Veri sabitleri, kullanılan veri saklama biçimlerini tarif eder;
- Hata ve uyarı kodları derleyici mesajlarını ve alım-satım sunucusunun alım-satım isteklerine cevaplarını tarif eder;
- Giriş/çıkış sabitleri, dosya fonksiyonları ile çalışmak ve MessageBox() fonksiyonu ile mesajları ekranda göstermek için tasarlanmıştır

Çizelge Sabitleri

Çizelgelerin çeşitli özelliklerini tanımlayan sabitler aşağıda belirtilen gruplara bölünmüştür:

- Olay tipleri – Çizelge ile çalışırken ortaya çıkan olaylar;
- Çizelge zaman dilimleri – standart kullanıma hazır periyotlar;
- Çizelge özellikleri – çizelge fonksiyonlarının parametreleri için kullanılan tanımlayıcılar;
- Konumlandırma sabitleri - ChartNavigate() fonksiyonunun parametre değeri;
- Çizelgelerin görüntülenmesi - çizelge görünümünün ayarlanması.

Çizelge Olaylarının Tipleri

OnChartEvent() fonksiyonu kullanılarak işlenebilecek 9 tip olay vardır. Özel olaylar için, CHARTEVENT_CUSTOM 'dan CHARTEVENT_CUSTOM_LAST 'a kadar 65535 adet dahili tanımlayıcı mevcuttur. Özel bir olay oluşturmak için EventChartCustom() fonksiyonu kullanılabilir.

ENUM_CHART_EVENT

Tanıtıcı	Açıklama
CHARTEVENT_KEYDOWN	Tuş darbeleri
CHARTEVENT_MOUSE_MOVE	Fare hareketi, fare tıklamaları (CHART_EVENT_MOUSE_MOVE=true, çizelge için ayarlanmışsa)
CHARTEVENT_MOUSE_WHEEL	Fare tekerleğinin tıklanması veya kaydırılması (çizelge için CHART_EVENT_MOUSE_WHEEL=True ise)
CHARTEVENT_OBJECT_CREATE	Grafiksel nesne oluşturma olayı (CHART_EVENT_OBJECT_CREATE=true, çizelge için ayarlanmışsa)
CHARTEVENT_OBJECT_CHANGE	Grafiksel nesnenin özellikler penceresinden değiştirilmesi olayı
CHARTEVENT_OBJECT_DELETE	Grafiksel nesnenin silinmesi olayı (CHART_EVENT_OBJECT_DELETE=true, çizelge için ayarlanmışsa)
CHARTEVENT_CLICK	Çizelge üstünde tıklama olayı
CHARTEVENT_OBJECT_CLICK	Grafiksel nesne üzerinde tıklama olayı
CHARTEVENT_OBJECT_DRAG	Grafiksel nesne üzerinde sürükleyip bırakma
CHARTEVENT_OBJECT_ENDEDIT	'Düzenle' grafiksel nesnesinde metin düzenleme işinin bitmesi
CHARTEVENT_CHART_CHANGE	Özellikler iletişim penceresi ile çizelge boyutunun veya çizelge özelliklerinin değişmesi
CHARTEVENT_CUSTOM	Bir özel olaylar dizisindeki ilk olayın numarası
CHARTEVENT_CUSTOM_LAST	Bir özel olaylar dizisindeki son olayın numarası

Her olay tipi için, OnChartEvent() fonksiyonunun giriş parametreleri, mevcut olayı işleyebilmek için kesin değerlere sahiptir. Bu parametreler ile geçirilen olaylar ve değerler aşağıdaki tabloda listelenmiştir.

Olay	id parametresinin değeri	lparam parametresinin değeri	dparam parametresinin değeri	sparam parametresinin değeri
Tuş darbesi olayı	CHARTEVENT_KEYDOWN	basılmış bir tuşun kodu	Tekrar sayısı (kullanıcının tuşa basılı tutması sonucu tekrarlanan tuş darbelerinin sayısı)	Klavye tuşlarının durumunu tarif eden bir bit maskesinin dizgi değeri
Fare olayları (CHART_EVENT_MOUSE_MOVE=true çizelge için ayarlanmışsa)	CHARTEVENT_MOUSE_MOVE	X koordinatı	Y koordinatı	Fare tuşlarının durumunu tarif eden bir bit maskesinin string tipli değeri
Fare tekerleği olayı (çizelge için CHART_EVENT_MOUSE_WHEEL= true ise)	CHARTEVENT_MOUSE_WHEEL	Fare düğmelerinin ve tuşların durumu, fare imlecinin X ve Y koordinatları hakkında bilgi veren bayraklar. Açıklama için alttaki örneğe bakınız	Fare tekerleğinin kaydırılması işlemi için delta değeri	—
Grafiksel nesne oluşturulma olayı (if CHART_EVENT_OBJECT_CREATE=true çizelge için ayarlanmışsa)	CHARTEVENT_OBJECT_CREATE	—	—	Oluşturulan grafiksel nesnenin adı
Özellikler iletişim kutusu ile bir nesnenin özelliğinin değiştirilmesi olayı	CHARTEVENT_OBJECT_CHANGE	—	—	Değiştirilen grafiksel nesnenin adı
Grafiksel nesnenin silinmesi olayı (Eğer çizelge için CHART_EVENT_OBJECT_DELETE=true şeklinde ayar yapılmışsa)	CHARTEVENT_OBJECT_DELETE	—	—	Silinen grafiksel nesnenin ismi

Olay	id parametresinin değeri	lparam parametresinin değeri	dparam parametresinin değeri	sparam parametresinin değeri
Çizelge üzerinde fare tıklaması olayı	CHARTEVENT_CLICK	X koordinatı	Y koordinatı	—
Çizelge üzerindeki bir grafiksel nesnenin fare ile tıklanması olayı	CHARTEVENT_OBJECT_CLICK	X koordinatı	Y koordinatı	Olayın gerçekleştiği grafiksel nesnenin ismi
Grafiksel nesnenin fare ile sürüklenmesi olayı	CHARTEVENT_OBJECT_DRAG	—	—	Taşınan grafiksel nesnenin adı
LabelEdit grafiksel nesnesinin giriş kutusunda yapılan metin düzenleme işinin bitmesi olayı	CHARTEVENT_OBJECT_ENDEDIT	—	—	Metin düzenlemesi tamamlanan LabelEdit grafiksel nesnesinin adı
Özellikler iletişim kutusu ile çizelge boyutunun veya özelliklerinin değişimi olayı	CHARTEVENT_CHART_CHANGE	—	—	—
N sayısının altındaki kullanıcı olayının kimliği	CHARTEVENT_CUSTOM+N	EventChartCustom() fonksiyonu tarafından ayarlanmış değer	EventChartCustom() fonksiyonu tarafından ayarlanmış değer	EventChartCustom() fonksiyonu tarafından ayarlanmış değer

Örnek:

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

```
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   Print(MQL5InfoString(MQL5_PROGRAM_NAME)," isimli uzman çalışıyor");
//--- nesne oluşturma olaylarını devreye sok
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
//--- nesne silme olaylarını devreye sok
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_DELETE,true);
//--- grafik özelliklerinin zorla güncellenmesi, olay işleme için hazırlık sağlar
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // Olay tanımlayıcı
                  const long& lparam,   // long tipi olay parametresi
                  const double& dparam, // double tipi olay parametresi
                  const string& sparam  // string tipi olay parametresi
                  )
  {
//--- çizelge üzerinde sol fare tuşu tıklandı
   if(id==CHARTEVENT_CLICK)
     {
      Print("Çizelge üzerindeki fare tıklamasının koordinatları: x = ",lparam,"  y = "
     }
//--- grafiksel nesne üzerinde fareye tıklandı
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      Print("fare '"+sparam+"' isimli nesne üzerinde tıklandı");
     }
//--- tuşa basıldı
   if(id==CHARTEVENT_KEYDOWN)
     {
      switch(lparam)
        {
         case KEY_NUMLOCK_LEFT:  Print("KEY_NUMLOCK_LEFT tuşuna basıldı");   break;
         case KEY_LEFT:          Print("KEY_LEFT tuşuna basıldı");           break;
         case KEY_NUMLOCK_UP:    Print("KEY_NUMLOCK_UP tuşuna basıldı");     break;
         case KEY_UP:            Print("KEY_UP tuşuna basıldı");             break;
         case KEY_NUMLOCK_RIGHT: Print("KEY_NUMLOCK_RIGHT tuşuna basıldı");  break;
         case KEY_RIGHT:         Print("KEY_RIGHT tuşuna basıldı");          break;
         case KEY_NUMLOCK_DOWN:  Print("KEY_NUMLOCK_DOWN tuşuna basıldı");   break;
         case KEY_DOWN:          Print("KEY_DOWN tuşuna basıldı");           break;
         case KEY_NUMPAD_5:      Print("KEY_NUMPAD_5 tuşuna basıldı");       break;
         case KEY_NUMLOCK_5:     Print("KEY_NUMLOCK_5 tuşuna basıldı");      break;
```

```
          default:                Print("listelenmemiş bir tuşa basıldı");
         }
       ChartRedraw();
      }
//--- nesne silindi
   if(id==CHARTEVENT_OBJECT_DELETE)
     {
      Print(sparam," isimli nesne silindi");
     }
//--- nesne oluşturuldı
   if(id==CHARTEVENT_OBJECT_CREATE)
     {
      Print(sparam," isimli nesne oluşturuldı");
     }
//--- nesne ya da tutturma noktasının koordinatları değiştirildi
   if(id==CHARTEVENT_OBJECT_DRAG)
     {
      Print(sparam," isimli nesnenin tutturma noktası koordinatları değiştirildi");
     }
//--- nesnenin Edit (düzenle) alanı içinde metni değişti
   if(id==CHARTEVENT_OBJECT_ENDEDIT)
     {
      Print(sparam," isimli nesnenin Edit alanındaki metni değiştirildi");
     }
  }
```

CHARTEVENT_MOUSE_MOVE olayı için **sparam** parametresi klavye ve fare tuşlarının durumu hakkında bilgi içerir:

Bit	Açıklama
1	Sol fare tuşunun durumu
2	Sağ fare tuşunun durumu
3	SHIFT tuşunun durumu
4	CTRL tuşunun durumu
5	Orta fare tuşunun durumu
6	İlk ek fare tuşunun durumu
7	İkinci ek fare tuşunun durumu

Örnek:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
void OnInit()
  {
```

```
//--- CHART_EVENT_MOUSE_MOVE mesajlarını devreye sok
   ChartSetInteger(0,CHART_EVENT_MOUSE_MOVE,1);
// --- grafiğin içerik menüsünü devre dışı bırakın (sağda)
   ChartSetInteger(0,CHART_CONTEXT_MENU,0);
// --- artı işaretini devre dışı bırak (orta düğmeden)
   ChartSetInteger(0,CHART_CROSSHAIR_TOOL,0);
//--- Grafik özelliklerinin zorla güncellenmesi, olay işleme için hazırlık sağlar
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| MouseState                                                       |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // fare sol
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // fare sağ
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // fare orta
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // fare ilk X tuşu
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // fare ikinci X tuşu
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // shift tuşu
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // control tuşu
   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));
  }
```

CHARTEVENT_MOUSE_WHEEL olayı için, **lparam** ve **dparam** parametreleri, **Ctrl** ve **Shift** tuşlarının, fare tuşlarının, imleç koordinatlarının ve fare tekerleğinin kaydırma değerinin durumu hakkında bilgi içerir. Daha iyi anlamak için bu uzman danışmanı bir çizelgede çalıştırıp, kodda yazılan tuşlara basarak fare tekerleğini kullanın.

CHARTEVENT_MOUSE_WHEEL olayının işlenmesi örneği:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
init OnInit()
  {
//--- Fare tekerleğinin kaydırılmasıyla ilgili mesajlar
   ChartSetInteger(0,CHART_EVENT_MOUSE_WHEEL,1);
//--- Grafik özelliklerinin zorla güncellenmesi, olay işleme için hazırlık sağlar
```

```
   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)
     {
      //--- Bu olay için fare tuşlarının ve tekerleğinin durumunu izleyin
      int flg_keys = (int)(lparam>>32);          // Ctrl, Shift ve fare tuşlarının du
      int x_cursor = (int)(short)lparam;         // fare tekerleği olayının gerçekleşt
      int y_cursor = (int)(short)(lparam>>16);   // fare tekerleği olayının gerçekleşt
      int delta    = (int)dparam;                // Fare tekerleği kaydırmasının topla
      //--- Bayrağın işlenmesi
      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_
     }
  }
//+------------------------------------------------------------------+ /*
   Çıktı örneği
   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  */
```

Ayrıca Bakınız

Olay İşleme Fonksiyonları, Olaylarla çalışmak

Çizelge Zaman-Aralıkları

Çizelgelerde kullanılan tüm zaman dilimleri benzersiz tanıtıcılara sahiptir. PERIOD_CURRENT tanıtıcısı MQL5 programının çalıştırıldığı çizelgenin mevcut periyodunu temsil eder.

ENUM_TIMEFRAMES

Tanıtıcı	Açıklama
PERIOD_CURRENT	Mevcut zaman aralığı
PERIOD_M1	1 dakika
PERIOD_M2	2 dakika
PERIOD_M3	3 dakika
PERIOD_M4	4 dakika
PERIOD_M5	5 dakika
PERIOD_M6	6 dakika
PERIOD_M10	10 dakika
PERIOD_M12	12 dakika
PERIOD_M15	15 dakika
PERIOD_M20	20 dakika
PERIOD_M30	30 dakika
PERIOD_H1	1 saat
PERIOD_H2	2 saat
PERIOD_H3	3 saat
PERIOD_H4	4 saat
PERIOD_H6	6 saat
PERIOD_H8	8 saat
PERIOD_H12	12 saat
PERIOD_D1	1 gün
PERIOD_W1	1 hafta
PERIOD_MN1	1 ay

Örnek:

```
   string chart_name="test_Object_Chart";
   Print(chart_name," isminde bir Chart (çizelge) nesnesi oluşturalım");
//--- Böyle bir nesne yoksa - oluştur
   if(ObjectFind(0,chart_name)<0)ObjectCreate(0,chart_name,OBJ_CHART,0,0,0,0,0);
```

```
//--- Sembolü tanımla
   ObjectSetString(0,chart_name,OBJPROP_SYMBOL,"EURUSD");
//--- Tutturma noktasının X koordinatını ayarla
   ObjectSetInteger(0,chart_name,OBJPROP_XDISTANCE,100);
//--- Tutturma noktasının Y koordinatını ayarla
   ObjectSetInteger(0,chart_name,OBJPROP_YDISTANCE,100);
//--- Çizelge genişliğini ayarla
   ObjectSetInteger(0,chart_name,OBJPROP_XSIZE,400);
//--- Yüksekliği ayarla
   ObjectSetInteger(0,chart_name,OBJPROP_YSIZE,300);
//--- Zaman aralığını ayarla
   ObjectSetInteger(0,chart_name,OBJPROP_PERIOD,PERIOD_D1);
//--- Ölçeği ayarla (0'dan 5'e kadar)
   ObjectSetDouble(0,chart_name,OBJPROP_SCALE,4);
//--- Fare ile seçimi devre dışı bırak
   ObjectSetInteger(0,chart_name,OBJPROP_SELECTABLE,false);
```

Zaman Serisi tanımlayıcısı

Zaman serisi tanımlayıcılarıiHighest() ve iLowest() fonksiyonlarında kullanılır. Sayımın değerine eşit olabilirler

ENUM_SERIESMODE

Tanımlayıcı	Açıklama
MODE_OPEN	Açılış fiyatı
MODE_LOW	Düşük fiyat
MODE_HIGH	Yüksek fiyat
MODE_CLOSE	Kapanış fiyatı
MODE_VOLUME	Tik Hacmi
MODE_REAL_VOLUME	Gerçek hacim
MODE_SPREAD	Spread

Ayrıca Bakınız

PeriodSeconds, Periyot, Tarih ve Zaman, Nesnelerin Görünürlüğü

Çizelge Özellikleri

ENUM_CHART_PROPERTY sayımının tanımlayıcıları çizelge çalışma fonksiyonlarının parametreleri şeklinde kullanılır. "Özellik tipi" sütunu içindeki r/o kısaltması bu özelliğin salt okunur olduğu ve değiştirilemeyeceği anlamına gelir. "Özellik tipi" sütunu içindeki w/o kısaltması bu özelliğin salt yazılır olduğu ve alınamayacağı anlamına gelir. Belli kesin özelliklere erişim gerçekleştirirken, çizelge alt pencerelerinin sayısını göstermeye yarayan fazladan bir parametre şekillendiricisi belirtilmelidir. '0' ana pencere anlamına gelir.

Çizelge özelliklerini tanımlayan fonksiyonlar aslında çizelgeye değişim komutları göndermek için kullanılır. Bu fonksiyonlar başarılı şekilde çalıştırılırsa, söz konusu değişim komutu çizelge olaylarının genel kuyruğuna eklenir. Çizelge olayları kuyruğu işlendiğinde değişimler çizelgeye uygulanır.

Bu nedenle, bu fonksiyonların çağrılmalarının hemen ardından çizelge üzerinde bir güncelleme beklememeniz gerekir. Çizelge genellikle değişim olaylarını takiben terminal tarafından otomatik olarak güncellenir (yeni bir fiyat teklifinin gelmesinden sonra, çizelge penceresinin yeniden boyutlandırılmasının ardından, vb). Çizelgeyi zorla güncellemek için ChartRedraw() fonksiyonunu kullanın.

ChartSetInteger() ve ChartGetInteger() fonksiyonları için

ENUM_CHART_PROPERTY_INTEGER

Tanıtıcı	Açıklama	Özellik Tipi
CHART_SHOW	Fiyat çizelgesi çizimi. 'false' ise, fiyat çizelgesine dair tüm bileşenlerin çizimi devre-dışı bırakılır ve çizelge sınırlarındaki bileşenler (zaman ve fiyat ölçekleri, hızlı gezinti çubuğu, takvim olaylarının etiketleri, işlem seviyeleri, gösterge ve çubuk ip-uçları, gösterge pencereleri, hacim histogramları, vb.) gösterilmez. Çizimi devre-dışı bırakmak, grafiksel kaynakların kullanımıyla özel program arayüzleri oluşturmak için mükemmel çözümdür. Grafiksel nesneler CHART_SHOW özelliğinin değerinden bağımsız olarak her zaman görüntülenirler.	bool
CHART_IS_OBJECT	"Çizelge" (OBJ_CHART) nesnesini tanımlama - bir grafiksel nesne için 'true' değerine döner. Gerçek bir çizelge için 'false' dönüşü yapar	bool r/o
CHART_BRING_TO_TOP	Çizelgeyi diğer çizelgelerin üstünde göster	bool
CHART_CONTEXT_MENU	Sağ fare tuşu ile içerik menüsüne erişimi devreye sokar/devre dışı bırakır.	bool (varsayılan değer: 'true')

Tanıtıcı	Açıklama	Özellik Tipi
	CHART_CONTEXT_MENU=false ise, sadece çizelgenin içerik menüsü devre dışı bırakılır. Çizelge nesnelerinin içerik menüsü etkin kalır.	
CHART_CROSSHAIR_TOOL	Orta fare tuşuyla artı imlecin çalıştırılmasını devreye sokar/devre dışı bırakır.	bool (varsayılan değer: 'true')
CHART_MOUSE_SCROLL	Sol fare tuşunu kullanarak çizelgeyi yatay olarak kaydırma. Eğer şu parametreler 'true' olarak ayarlanmışsa dikey kaydırma da mümkündür: CHART_SCALEFIX, CHART_SCALEFIX_11 veya CHART_SCALE_PT_PER_BAR CHART_MOUSE_SCROLL=false ise, fare tekerleği ile kaydırma devre dışı bırakılır	bool
CHART_EVENT_MOUSE_WHEEL	Fare tekerleğinin olaylarıyla (CHARTEVENT_MOUSE_WHEEL) ilgili tüm mql5 programlarına mesaj gönderir.	bool (varsayılan değer: 'true')
CHART_EVENT_MOUSE_MOVE	Yapılan fare hareketlerini ve tıklamalarını (CHARTEVENT_MOUSE_MOVE) çizelge üzerindeki tüm MQL5 programlarına gönderir	bool
CHART_EVENT_OBJECT_CREATE	Yeni nesne oluşturulması olayının bildirilerini (CHARTEVENT_OBJECT_CREATE) çizelge üzerindeki tüm MQL5 programlarına gönderir	bool
CHART_EVENT_OBJECT_DELETE	Nesne silinmesi olayını (CHARTEVENT_OBJECT_DELETE) çizelge üzerindeki tüm MQL5 programlarına gönderir	bool
CHART_MODE	Çizelge tipi (mumlar, çubuklar veya çizgiler)	enum ENUM_CHART_MODE
CHART_FOREGROUND	Ön plandaki fiyat çizelgesi	bool
CHART_SHIFT	Fiyat çizelgesinin sağ sınırdan girintisinin modu	bool
CHART_AUTOSCROLL	Çizelgenin sağ sınırına otomatik hareket etme modu	bool
CHART_KEYBOARD_CONTROL	Çizelgenin klavyeden yönetilmesine izin ver ("Home", "End", "PageUp", "+", "-", "yukarı ok", vb.). CHART_KEYBOARD_CONTROL değerinin	bool

Tanıtıcı	Açıklama	Özellik Tipi
	'false' yapılması, çizelgenin kaydırılmasını ve ölçeklendirilmesini devre dışı bırakır ama OnChartEvent() dahilindeki klavye olaylarının alınması etkilenmez.	
CHART_QUICK_NAVIGATION	Hızlı gezinti çubuğunu etkinleştirmek için çizelgenin Space ve Enter tuşlarını yakalamasına izin verin. Hızlı gezinti çubuğu fare tuşunun çift tıklanması veya Space/Enter tuşlarının kullanılması durumunda otomatik olarak açılır. Sembolün, zaman diliminin ve ilk görünür çubuğun tarihinin kolayca değiştirilmesini sağlar.	bool
CHART_SCALE	Ölçek	int 0'dan 5'e kadar
CHART_SCALEFIX	Sabit ölçek modu	bool
CHART_SCALEFIX_11	1:1 ölçek modu	bool
CHART_SCALE_PT_PER_BAR	Çubuk başına düşen puan bazında belirlenen ölçek	bool
CHART_SHOW_TICKER	Sol üst köşede sembol adını görüntüler. CHART_SHOW_TICKER'ı 'false' olarak ayarlamak CHART_SHOW_OHLC'yi de 'false' olarak ayarlar ve OHLC'yi devre dışı bırakır	bool
CHART_SHOW_OHLC	Sol üst köşede OHLC değerlerini görüntüler. CHART_SHOW_OHLC'yi 'true' olarak ayarlamak CHART_SHOW_TICKER'ı da 'true' olarak ayarlar ve sembol adının görüntülenmesini etkinleştirir	bool
CHART_SHOW_BID_LINE	Alış değerlerini çizelgede bir yatay çizgi olarak görüntüle	bool
CHART_SHOW_ASK_LINE	Satış değerlerini çizelgede bir yatay çizgi olarak görüntüle	bool
CHART_SHOW_LAST_LINE	Son değerleri çizelgede bir yatay çizgi olarak görüntüle	bool
CHART_SHOW_PERIOD_SEP	Bitişik periyotlar arsında dikey ayraçlar görüntüle	bool
CHART_SHOW_GRID	Çizelgede ızgarayı görüntüle	bool
CHART_SHOW_VOLUMES	Çizelgede hacimi görüntüle	enum ENUM_CHART_VOLUME_MODE

Tanıtıcı	Açıklama	Özellik Tipi
CHART_SHOW_OBJECT_DESCR	Grafiksel nesnelerin kendiliğinden açılan (pop-up) tarifleri	bool
CHART_SHOW_TRADE_HISTORY	İşlem geçmişindeki ticaret işlemlerini grafikte giriş/çıkış oklarıyla görüntüler. Platform ayarlarından "İşlem geçmişini göster" seçeneğinin açıklamalarına bakın.	bool
CHART_VISIBLE_BARS	Çizelge üstündeki görüntülenebilir çubukların sayısı	int r/o
CHART_WINDOWS_TOTAL	Gösterge alt pencereleri de dahil olmak üzere, toplam çizelge penceresi sayısı	int r/o
CHART_WINDOW_IS_VISIBLE	Alt pencerelerin görünürlüğü	bool r/o şekillendirici - alt pencere sayısı
CHART_WINDOW_HANDLE	Çizelge penceresi işleyicisi (HWND)	int r/o
CHART_WINDOW_YDISTANCE	Dikey Y ekseni üzerinde piksel bazında, gösterge alt penceresinin üst çerçevesi ile ana çizelge üst çerçevesi arasındaki uzaklık. Bir fare olayı durumunda imleç koordinatları ana çizelge penceresinin koordinatları şeklinde geçirilir. Gösterge alt pencerelerindeki grafiksel nesnelerin koordinatları ise alt pencerenin sol üst köşesine göre ayarlanır. Bu değer, koordinatları alt pencerenin sol üst köşesine göre ayarlanan grafiksel nesnelerle doğru şekilde çalışmak amacıyla, ana çizelgenin kesin koordinatlarını alt pencerenin yerel koordinatlarına dönüştürmek için gereklidir.	int r/o şekillendirici - alt pencere numarası
CHART_FIRST_VISIBLE_BAR	Çizelgedeki görünür ilk çubuğun numarası. Çubukların indislenmesi zaman serilerinde olduğu gibidir.	int r/o
CHART_WIDTH_IN_BARS	Çubuk sayısıyla çizelge genişliği	int r/o
CHART_WIDTH_IN_PIXELS	Piksel bazında çizelge genişliği	int r/o
CHART_HEIGHT_IN_PIXELS	Piksel bazında çizelge yüksekliği	int şekillendirici - alt pencere numarası
CHART_COLOR_BACKGROUND	Çizelge arka plan rengi	color

Tanıtıcı	Açıklama	Özellik Tipi
CHART_COLOR_FOREGROUND	Eksenlerin, ölçeklerin ve OHLC çizgisinin rengi	color
CHART_COLOR_GRID	Izgara rengi	color
CHART_COLOR_VOLUME	Hacimlerin ve pozisyon açma seviyelerinin rengi	color
CHART_COLOR_CHART_UP	Yükseliş çubuğunun, boğa mumunun gölgesinin ve gövde sınırlarının rengi	color
CHART_COLOR_CHART_DOWN	Düşüş çubuğunun, ayı mumunun gölgesinin ve gövde sınırlarının rengi	color
CHART_COLOR_CHART_LINE	Çizgi çizelgesi rengi ve Japon "Doji" mumları	color
CHART_COLOR_CANDLE_BULL	Boğa mumunun gövde rengi	color
CHART_COLOR_CANDLE_BEAR	Ayı mumunun gövde rengi	color
CHART_COLOR_BID	Satış fiyatı seviyesi rengi	color
CHART_COLOR_ASK	Satış fiyatı seviyesi rengi	color
CHART_COLOR_LAST	Son gerçekleşen fiyatın rengi	color
CHART_COLOR_STOP_LEVEL	Stop (durdurma) emri seviyelerinin rengi (Zarar Durdur ve Kar Al)	color
CHART_SHOW_TRADE_LEVELS	Alım-satım seviyelerinin çizelgede gösterilmesi (açık pozisyonların, Zarar Durdur, Kar Al ve bekleyen emirlerin seviyeleri)	bool
CHART_DRAG_TRADE_LEVELS	Alım-satım seviyelerinin çizelge üzerinde fare ile sürüklenmesi izni. Sürükleme modu varsayılan olarak devrededir (true değeri)	bool
CHART_SHOW_DATE_SCALE	Çizelge üzerinde zaman ölçeğinin gösterilmesi	bool
CHART_SHOW_PRICE_SCALE	Çizelge üzerinde fiyat ölçeğinin gösterilmesi	bool
CHART_SHOW_ONE_CLICK	Çizelge üzerinde "Tek Tıkla İşlem"panelinin gösterilmesi	bool
CHART_IS_MAXIMIZED	Çizelge penceresi maksimize edildi	bool r/o
CHART_IS_MINIMIZED	Çizelge penceresi minimize edildi	bool r/o

Tanıtıcı	Açıklama	Özellik Tipi
CHART_IS_DOCKED	Grafik penceresi yerleşiktir. Eğer false olarak ayarlanırsa, grafik, terminal alanı dışına sürüklenebilir	bool
CHART_FLOAT_LEFT	Yerleşik olmayan grafik penceresinin sanal ekrana göre sol koordinatı	int
CHART_FLOAT_TOP	Yerleşik olmayan grafik penceresinin sanal ekrana göre üst koordinatı	int
CHART_FLOAT_RIGHT	Yerleşik olmayan grafik penceresinin sanal ekrana göre sağ koordinatı	int
CHART_FLOAT_BOTTOM	Yerleşik olmayan grafik penceresinin sanal ekrana göre alt koordinatı	int

ChartSetDouble() ve ChartGetDouble() fonksiyonları için

ENUM_CHART_PROPERTY_DOUBLE

Tanıtıcı	Açıklama	Özellik Tipi
CHART_SHIFT_SIZE	Sıfır çubuğunun sağ sınırdan olan girintisinin yüzdelik değeri	double (yüzde 10'dan 50'ye kadar)
CHART_FIXED_POSITION	yüzdelik değerle, sol sınırdan sabit çizelge konumu. Sabit çizelge pozisyonu yatay zaman ekseni üzerinde küçük gri bir üçgen ile belirtilir. Bu üçgen sadece yeni tik anında çizelgenin otomatik kaydırılması devre dışı bırakılmışsa görüntülenir (CHART_AUTOSCROLL özelliğine bakın). Sabit pozisyon üzerindeki çubuklar yakınlaştırma ve	double

Tanıtıcı	Açıklama	Özellik Tipi
	uzaklaştırma sırasında aynı yerde kalır.	
CHART_FIXED_MAX	Sabitlenmiş çizelge maksimumu	double
CHART_FIXED_MIN	Sabitlenmiş çizelge minimumu	double
CHART_POINTS_PER_BAR	Çubuk başına düşen puan bazında ölçek	double
CHART_PRICE_MIN	Çizelge minimumu	double r/o şekillendirici - alt pencere numarası
CHART_PRICE_MAX	Çizelge maksimumu	double r/o şekillendirici - alt pencere numarası

ChartSetString() ve ChartGetString() fonksiyonları için

ENUM_CHART_PROPERTY_STRING

Tanıtıcı	Açıklama	Özellik Tipi
CHART_COMMENT	Çizelgedeki yorumun metni	string
CHART_EXPERT_NAME	The name of the Expert Advisor running on the chart with the specified chart_id	string r/o
CHART_SCRIPT_NAME	The name of the script running on the chart with the specified chart_id	string r/o

Örnek:

```
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," piksel");
      Print(i+": CHART_PRICE_MIN = ",priceMin);
      Print(i+": CHART_PRICE_MAX = ",priceMax);
     }
  }
```

Ayrıca Bakınız

Çizelgeyle Çalışma Örnekleri

Konumlandırma Sabitleri

ENUM_CHART_POSITION listesinden üç tanımlayıcı, ChartNavigate() fonksiyonunun *position* parametresinin muhtemel değerleridir.

ENUM_CHART_POSITION

Tanıtıcı	Açıklama
CHART_BEGIN	Çizelge başlangıcı (en eski fiyat)
CHART_CURRENT_POS	Şimdiki konum
CHART_END	Çizelge sonu (en son fiyat)

Örnek:

```
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("Konumlandırma başarısız. Hata = ",GetLastError());
   ChartRedraw();
  }
```

Çizelge Sunumu

Fiyat çizelgeleri üç şekilde görüntülenebilir:

- çubuklar şeklinde;
- mumlar şeklinde;
- çizgi şeklinde.

Fiyat grafiğini görüntülemenin özel bir yolu ChartSetInteger(chart_handle,CHART_MODE, chart_mode) fonksiyonu ile ayarlanır, burada chart_mode, ENUM_CHART_MODE sayımının değerlerinden biridir.

ENUM_CHART_MODE

Tanıtıcı	Açıklama
CHART_BARS	Çubuk dizilimi şeklinde gösterir
CHART_CANDLES	Japon mumları şeklinde gösterir
CHART_LINE	Kapanış fiyatları ile çizilen bir çizgi şeklinde gösterir

Çizelgede hacimlerinin gösterim modunu belirlemek için ChartSetInteger(çizelge_tanıtıcısı, CHART_SHOW_VOLUMES, hacim_modu) fonksiyonu kullanılır, burada hacim_modu, ENUM_CHART_VOLUME_MODE sayımının değerlerinden biridir.

ENUM_CHART_VOLUME_MODE

Tanıtıcı	Açıklama
CHART_VOLUME_HIDE	Hacimler gösterilmez
CHART_VOLUME_TICK	Tik Hacimleri
CHART_VOLUME_REAL	Alım-satım hacimleri

Örnek:

```
//--- Mevcut çizelgenin işleyicisini al
   long handle=ChartID();
   if(handle>0) // Eğer başarılı olmuşsa, ayrıyeten özelleştir
     {
      //--- Otomatik kaydırmayı devre dışı bırak
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- Çizelgenin sağ sınırının girintisini ayarla
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- mumlar şeklinde görüntüle
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- Tarih başlangıcından itibaren 100 çubuk kaydır
      ChartNavigate(handle,CHART_CURRENT_POS,100);
      //--- Tik hacmi görüntü modunu ayarla
      ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);
```

```
   }
```

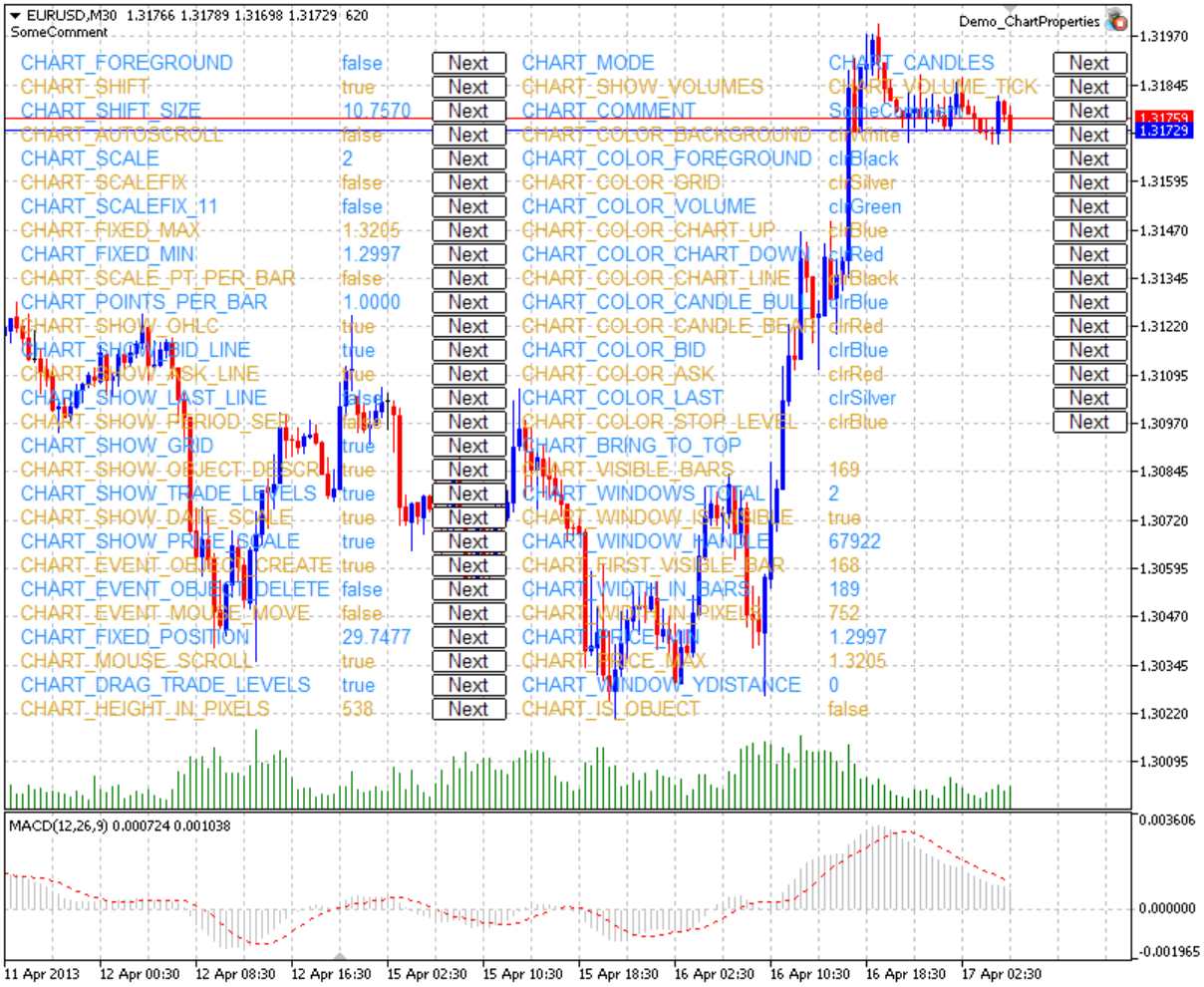
Ayrıca Bakınız

ChartOpen, ChartID

Çizelgeyle Çalışma Örnekleri

Bu bölüm çizelge özellikleriyle çalışma örnekleri içermektedir. Her özellik için bir veya iki fonksiyon gösterilmiştir. Bu fonksiyonlar özellik değerini ayarlama/alma olanağı sağlar. Bu fonksiyonlar "olduğu gibi" özel MQL5 uygulamalarında kullanılabilirler.

Aşağıdaki ekran görüntüsü çizelge özelliğini değiştirmenin görünümü nasıl değiştireceğini sergileyen grafik panelini göstermektedir. İleri düğmesine basarak özelliğin yeni değerini ayarlayabilir ve değişimleri çizelge penceresinde görüntüleyebilirsiniz.



Panelin kaynak kodu aşağıda yer almaktadır.

Çizelge özellikleri ve bunlarla çalışmak için örnek fonksiyonlar

- **CHART_IS_OBJECT** bir nesnenin gerçek bir çizelge mi yoksa bir grafik nesnesi mi olduğunu belirtir

```
//+------------------------------------------------------------------+
//| Bir nesnenin çizelge olup olmadığını tanımlar. Eğer bir          |
//| grafik nesnesiyse, sonuç true olur. Eğer gerçek bir              |
//| çizelge ise sonuç değişkeni false değerine sahip olur.           |
//+------------------------------------------------------------------+
bool ChartIsObject(bool &result,const long chart_ID=0)
  {
```

```
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- çizelge özelliğini al
   if(!ChartGetInteger(chart_ID,CHART_IS_OBJECT,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      //--- false dönüşü yap
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_BRING_TO_TOP**, çizelgeyi tüm diğerlerinin üstünde gösterir.

```
//+------------------------------------------------------------------------------+
//| Çizelgeyi tüm diğerlerinin üstünde göstermek için terminale komut yolla.  |
//+------------------------------------------------------------------------------+
bool ChartBringToTop(const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- çizelgeyi diğer hepsinin üstünde göster
   if(!ChartSetInteger(chart_ID,CHART_BRING_TO_TOP,0,true))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_MOUSE_SCROLL**, sol fare tuşun ile çizelgeyi kaydırmak için kullanılan bir özelliktir.

```
//+------------------------------------------------------------------------------+
//| Fonksiyon, sol fare tuşuyla grafiği kaydırma özelliğinin aktif olup
//| olmadığını tanımlar.
//+------------------------------------------------------------------------------+
bool ChartMouseScrollGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
```

```
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, sol fare tuşuyla çizelge kaydırmayı etkinleştirir/devre |
//| dışı bırakır.                                                    |
//+------------------------------------------------------------------+
bool ChartMouseScrollSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_MOUSE_SCROLL,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_EVENT_MOUSE_MOVE,** taşıma olayları ve MQL5 uygulamalarına yapılan fare tıklamaları ile ilgili mesajlar gönderen bir özelliktir (CHARTEVENT_MOUSE_MOVE).

```
//+------------------------------------------------------------------+
//| Taşıma olayları ve fare tıklamalarıyla ilgili mesajlar           |
//| çizelgedeki tüm MQL5 uygulamalarına gönderildi mi kontrol et.    |
//+------------------------------------------------------------------+
bool ChartEventMouseMoveGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
```

```
   if(!ChartGetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Fonksiyon, taşıma olayları ve çizelge üzerindeki MQL5 uygulamalarına yapılan |
//| fare tıklamalarına ilişkin mesajların gönderim modunu etkinleştirir veya     |
//| devre dışı bırakır.                                                          |
//+------------------------------------------------------------------------------+
bool ChartEventMouseMoveSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_EVENT_MOUSE_MOVE,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_EVENT_OBJECT_CREATE**, grafik nesnesi oluşturma olayı (CHARTEVENT_OBJECT_CREATE) ile ilgili mesajlar gönderme özelliğidir.

```
//+------------------------------------------------------------------+
//| Grafik nesnesinin oluşturulması olayı ile ilgili mesajlar        |
//| çizelgedeki tüm MQL5 uygulamalarına gönderildi mi kontrol et.    |
//+------------------------------------------------------------------+
bool ChartEventObjectCreateGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
```

```
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------------+
//| Fonksiyon, çizelge üzerindeki tüm MQL5 uygulamalarına grafik nesnesinin  |
//| oluşturulması olayı ile ilgili mesaj gönderme modunu devreye sokar veya    |
//| devre dışı bırakır.                                                      |
//+------------------------------------------------------------------------+
bool ChartEventObjectCreateSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_CREATE,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_EVENT_OBJECT_DELETE**, MQL5 uygulamalarına, grafik nesnelerinin silinmesi olayı (CHARTEVENT_OBJECT_DELETE) ile ilgili mesaj gönderme özelliğidir.

```
//+------------------------------------------------------------------+
//| Grafik nesnesinin silinme olayı ile ilgili mesajlar              |
//| çizelgedeki tüm MQL5 uygulamalarına gönderildi mi kontrol et.    |
//+------------------------------------------------------------------+
bool ChartEventObjectDeleteGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
```

```
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------------+
//| Fonksiyon, çizelge üzerindeki tüm MQL5 uygulamalarına grafik nesnesinin  |
//| silinme olayı ile ilgili mesaj gönderimini devreye sokar veya            |
//| devre dışı bırakır.                                                      |
//+------------------------------------------------------------------------+
bool ChartEventObjectDeleteSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_EVENT_OBJECT_DELETE,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_MODE** - çizelge tipi (mumlar çubuklar veya çizgi).

```
//+------------------------------------------------------------------+
//| Çizelge görüntüleme tipini al (mumlar çubuklar veya              |
//| çizgi).                                                          |
//+------------------------------------------------------------------+
ENUM_CHART_MODE ChartModeGet(const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long result=WRONG_VALUE;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_MODE,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((ENUM_CHART_MODE)result);
  }
//+------------------------------------------------------------------+
//| Çizelge görüntüleme tipini ayarla (mumlar, çubuklar veya         |
```

```
//| çizgi).                                                          |
//+------------------------------------------------------------------+
bool ChartModeSet(const long value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_MODE,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_FOREGROUND**, bir fiyat çizelgesini ön planda gösterme özelliğidir.

```
//+------------------------------------------------------------------+
//| Fonksiyon, fiyat çizelgesi ön planda gösterilmiş mi,             |
//| tanımla.                                                         |
//+------------------------------------------------------------------+
bool ChartForegroundGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------------+
//| Fonksiyon, fiyat çizelgesinin ön planda gösterimini etkinleştirir veya |
//| tanımla.                                                               |
//+------------------------------------------------------------------------+
bool ChartForegroundSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
```

```
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_FOREGROUND,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_SHIFT** - fiyat çizelgesinin sağ sınırdan itibaren kaydırılma modu.

```
//+------------------------------------------------------------------------------------+
//| Fonksiyon, fiyat çizelgesinin sağ sınırdan itibaren kaydırılma modu
//| devrede olup olmadığını tanımlar.
//+------------------------------------------------------------------------------------+
bool ChartShiftGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_SHIFT,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Fonksiyon, sağ sınırdan kaydırılmış görüntü bir fiyat çizelgesinin       |
//| görüntüleme modunu etkinleştirir/devre dışı bırakır.                     |
//+------------------------------------------------------------------------------------+
bool ChartShiftSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_SHIFT,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
```

```
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_AUTOSCROLL** - çizelgenin sağ sınırına otomatik kaydırma modu.

```
//+------------------------------------------------------------------+
//| Fonksiyon, yeni tik alınması durumunda, çizelgenin sağa doğru    |
//| otomatik kaydırma modunun devrede olup olmadığını tanımlar.      |
//+------------------------------------------------------------------+
bool ChartAutoscrollGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, yeni tik alınması durumunda, çizelgenin sağa doğru    |
//| otomatik kaydırılması modunu devreye sokar/devre dışı bırakır.   |
//+------------------------------------------------------------------+
bool ChartAutoscrollSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_AUTOSCROLL,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
```

```
  }
```

- **CHART_SCALE** - çizelge ölçek özelliği.

```
//+------------------------------------------------------------------+
//| Çizelge ölçeğini al (0'dan 5'e).                                 |
//+------------------------------------------------------------------+
int ChartScaleGet(const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long result=-1;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_SCALE,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((int)result);
  }
//+------------------------------------------------------------------+
//| Çizelge ölçeğini ayarla (0'dan 5'e).                             |
//+------------------------------------------------------------------+
bool ChartScaleSet(const long value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_SCALE,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_SCALEFIX** - sabit çizelge ölçeği modu.

```
//+------------------------------------------------------------------+
//| Fonksiyon, sabit ölçeği modu devrede mi, tanımlar.               |
//+------------------------------------------------------------------+
bool ChartScaleFixGet(bool &result,const long chart_ID=0)
  {
```

```
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fonksiyon sabit ölçek modunu devreye sokar/devre dışı bırakır.   |
//+------------------------------------------------------------------+
bool ChartScaleFixSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_SCALEFIX_11** - 1:1 çizelge ölçeği modu.

```
//+------------------------------------------------------------------+
//| Fonksiyon, "1:1" ölçeği devrede mi, tanımlar.                    |
//+------------------------------------------------------------------+
bool ChartScaleFix11Get(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
     {
```

```
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, "1:1" ölçek modunu etkinleştirir/devre dışı bırakır   |
//+------------------------------------------------------------------+
bool ChartScaleFix11Set(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_SCALEFIX_11,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_SCALE_PT_PER_BAR** - çizelge ölçeğini çubuk başına düşen puan bazında tanımlama modu.

```
//+------------------------------------------------------------------------------+
//| Fonksiyon, çizelge ölçeğini çubuk başına düşen puan bazında tanımlama        |
//| mi, değil mi tanımlar.                                                       |
//+------------------------------------------------------------------------------+
bool ChartScalePerBarGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
```

```
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------------------------------+
//| Fonksiyon, çizelge ölçeğini çubuk başına düşen puan bazında tanımlama modunu
//| deveye sokar/devre dışı bırakır.
//+------------------------------------------------------------------------------------------+
bool ChartScalePerBarSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_SCALE_PT_PER_BAR,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_SHOW_OHLC** - sol üst köşede OHLC değerlerinin gösterilmesi özelliği.

```
//+------------------------------------------------------------------+
//| Fonksiyon, sol üst köşede OHLC değerlerinin gösterilmesi         |
//| özelliği devrede mi değil mi tanımlar.                           |
//+------------------------------------------------------------------+
bool ChartShowOHLCGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Fonksiyon, sol üst köşede OHLC değerlerinin gösterilmesi modunu          |
```

```
//| devreye sokar/devre dışı bırakır.                                |
//+------------------------------------------------------------------+
bool ChartShowOHLCSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OHLC,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_SHOW_BID_LINE** - Satış değerini çizelge üzerinde yatay çizgiyle gösterme özelliği.

```
//+------------------------------------------------------------------+
//| Fonksiyon satış değeri çizgisinin çizelge üzerinde gösterilme modunun |
//| devrede olup olmadığını tanımlar.                                |
//+------------------------------------------------------------------+
bool ChartShowBidLineGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, Satış çizgisinin görüntülenme modunu devreye sokar veya |
//| devre dışı bırakır.                                              |
//+------------------------------------------------------------------+
bool ChartShowBidLineSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
```

```
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_SHOW_BID_LINE,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_SHOW_ASK_LINE** - Alış değerini çizelge üzerinde yatay çizgiyle gösterme özelliği.

```
//+------------------------------------------------------------------+
//| Fonksiyon Alış değeri çizgisinin çizelge üzerinde gösterilme modunun  |
//| devre dışı bırakır.                                              |
//+------------------------------------------------------------------+
bool ChartShowAskLineGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fonksiyon alış değeri çizgisinin çizelge üzerinde gösterilme modunu   |
//| devre dışı bırakır.                                              |
//+------------------------------------------------------------------+
bool ChartShowAskLineSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ASK_LINE,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
```

```
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_SHOW_LAST_LINE** - Son değeri çizelge üzerinde yatay çizgiyle gösterme özelliği.

```
//+------------------------------------------------------------------------------------+
//| Fonksiyon, son işlem fiyatı çizgisinin gösterilme modu                             |
//| devrede mi değil mi tanımlar.                                                      |
//+------------------------------------------------------------------------------------+
bool ChartShowLastLineGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------------------------+
//| Fonksiyon, gerçekleştirilen son işlemin fiyat çizgisinin gösterilme modunu         |
//| devreye sokar/devre dışı bırakır.                                                  |
//+------------------------------------------------------------------------------------+
bool ChartShowLastLineSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_SHOW_LAST_LINE,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
```

```
  }
```

- **CHART_SHOW_PERIOD_SEP** - bitişik periyotlar arası dikey ayraçları gösterme özelliği.

```
//+------------------------------------------------------------------+
//| Fonksiyon bitişik periyotlar arası dikey ayraçların gösterim     |
//| modunun devrede olup olmadığını tanımlar.                        |
//+------------------------------------------------------------------+
bool ChartShowPeriodSeparatorGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| fonksiyon, bitişik periyotlar arası dikey ayraçların görüntüleme |
//| modunu devreye sokar/devre dışı bırakır.                         |
//+------------------------------------------------------------------+
bool ChartShowPeriodSepapatorSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PERIOD_SEP,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_SHOW_GRID** - çizelge ızgarasının görüntülenme modu.

```
//+------------------------------------------------------------------+
//| Fonksiyon, çizelge ızgarası devrede mi değil mi tanımlar.        |
//+------------------------------------------------------------------+
bool ChartShowGridGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_SHOW_GRID,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fonksiyon çizelge ızgarasını devreye sokar/devre dışı bırakır.   |
//+------------------------------------------------------------------+
bool ChartShowGridSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_SHOW_GRID,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_SHOW_VOLUMES** - çizelge üzerinde hacimleri gösterme özelliği.

```
//+------------------------------------------------------------------------+
//| Fonksiyon, hacimler çizelgede gösteriliyor mu (gösterilmiyor mu,       |
//| tik hacimleri mi, gerçek hacimler mi gösteriliyor, tanımlar).          |
//+------------------------------------------------------------------------+
ENUM_CHART_VOLUME_MODE ChartShowVolumesGet(const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
```

```
   long result=WRONG_VALUE;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_SHOW_VOLUMES,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((ENUM_CHART_VOLUME_MODE)result);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, çizelge üzerinde hacimlerin gösterim modunu ayarlar.  |
//+------------------------------------------------------------------+
bool ChartShowVolumesSet(const long value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_SHOW_VOLUMES,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_SHOW_OBJECT_DESCR** - grafiksel nesnenin kendiliğinden-açılan (pop-up) tarifinin özelliği.

```
//+------------------------------------------------------------------+
//| Fonksiyon, fare ile üzerine gelindiğinde grafiksel nesnenin      |
//| pop-up tarifinin gözüküp gözükmediğini tanımlar.                 |
//+------------------------------------------------------------------+
bool ChartShowObjectDescriptionGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
```

```
      }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------------+
//| Fonksiyon, fare ile üzerine gelindiğinde grafiksel nesnenin pop-up     |
//| tarifinin gösterim modunu etkinleştirir/devre dışı bırakır.            |
//+------------------------------------------------------------------------+
bool ChartShowObjectDescriptionSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_SHOW_OBJECT_DESCR,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_VISIBLE_BARS**, çizelge üzerinde görüntüleme için kullanılabilecek çubuk sayısını tanımlar.

```
//+------------------------------------------------------------------------+
//| Fonksiyon, çizelge penceresinde görüntülenen çubuk sayısını alır       |
//+------------------------------------------------------------------------+
int ChartVisibleBars(const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long result=-1;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_VISIBLE_BARS,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((int)result);
  }
```

- **CHART_WINDOWS_TOTAL**, gösterge alt pencereleri de dahil olmak üzere, çizelge pencerelerinin toplam sayısını tanımlar.

```
//+------------------------------------------------------------------------+
//| Fonksiyon, gösterge alt pencereleri de dahil olmak üzere, çizelge      |
//| pencerelerinin toplam sayısını tanımlar.                               |
//+------------------------------------------------------------------------+
int ChartWindowsTotal(const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long result=-1;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_WINDOWS_TOTAL,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((int)result);
  }
```

- **CHART_WINDOW_IS_VISIBLE**, alt pencere görünürlüğünü tanımlar.

```
//+------------------------------------------------------------------+
//| Fonksiyon, mevcut çizelge penceresinin veya alt pencerenin       |
//| görünürlüğünü tanılar.                                           |
//+------------------------------------------------------------------+
bool ChartWindowsIsVisible(bool &result,const long chart_ID=0,const int sub_window=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_IS_VISIBLE,sub_window,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_WINDOW_HANDLE** çizelge işleyicisine dönüş yapar.

```
//+------------------------------------------------------------------+
//| Fonksiyon, çizelge işleyicisini alır                             |
//+------------------------------------------------------------------+
int ChartWindowsHandle(const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long result=-1;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_HANDLE,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((int)result);
  }
```

- **CHART_WINDOW_YDISTANCE**, gösterge penceresinin üst çerçevesi ile ana çizelge penceresinin üst çerçevesi arasındaki uzaklığı piksel bazında tanımlar.

```
//+------------------------------------------------------------------+
//| Fonksiyon, gösterge penceresinin üst çerçevesi ile ana çizelge   |
//| üst çerçevesi arasındaki uzaklığı piksel bazında alır.           |
//+------------------------------------------------------------------+
int ChartWindowsYDistance(const long chart_ID=0,const int sub_window=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long result=-1;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_WINDOW_YDISTANCE,sub_window,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((int)result);
  }
```

- **CHART_FIRST_VISIBLE_BAR**, çizelgede görünen ilk çubuğun numarasına dönüş yapar (çubukların indislenmesi zaman serilerine) karşılık gelir.

```
//+------------------------------------------------------------------------------------+
//| Fonksiyon, çizelgede görünen ilk çubuğun numarasını alır.
//| İndisleme zaman serilerindeki gibidir, son çubuklar daha küçük indislere sahiptir
//+------------------------------------------------------------------------------------+
int ChartFirstVisibleBar(const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long result=-1;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_FIRST_VISIBLE_BAR,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((int)result);
  }
```

- **CHART_WIDTH_IN_BARS**, çubuk bazında çizelge genişliğine dönüş yapar.

```
//+------------------------------------------------------------------+
//| Fonksiyon, çubuk bazında çizelge genişliğini alır.               |
//+------------------------------------------------------------------+
int ChartWidthInBars(const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long result=-1;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_BARS,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((int)result);
  }
```

- **CHART_WIDTH_IN_PIXELS**, piksel bazında çizelge genişliğine dönüş yapar.

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

```
//| Fonksiyon, piksel bazında çizelge genişliğini alır.              |
//+------------------------------------------------------------------+
int ChartWidthInPixels(const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long result=-1;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_WIDTH_IN_PIXELS,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((int)result);
  }
```

- **CHART_HEIGHT_IN_PIXELS** - piksel bazında çizelge yüksekliği özelliği.

```
//+------------------------------------------------------------------+
//| Fonksiyon, piksel bazında çizelge yüksekliği değerini alır.      |
//+------------------------------------------------------------------+
int ChartHeightInPixelsGet(const long chart_ID=0,const int sub_window=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long result=-1;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((int)result);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, piksel bazında çizelge yüksekliğini ayarlar.          |
//+------------------------------------------------------------------+
bool ChartHeightInPixelsSet(const int value,const long chart_ID=0,const int sub_windo
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_HEIGHT_IN_PIXELS,sub_window,value))
     {
```

```
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_COLOR_BACKGROUND** - çizelge arka plan rengi.

```
//+------------------------------------------------------------------+
//| Fonksiyon çizelge arka plan rengini alır.                        |
//+------------------------------------------------------------------+
color ChartBackColorGet(const long chart_ID=0)
  {
//--- rengi almak için değişkeni hazırla
   long result=clrNONE;
//--- hata değerini sıfırla
   ResetLastError();
//--- çizelge arka plan rengini al
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BACKGROUND,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, çizelge arka plan rengini ayarlar.                    |
//+------------------------------------------------------------------+
bool ChartBackColorSet(const color clr,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- çizelge arka plan rengini ayarla
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BACKGROUND,clr))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_COLOR_FOREGROUND** - eksenin, ölçeğin ve OHLC çizgisinin rengi.

```
//+------------------------------------------------------------------+
//| Fonksyon, eksenin, ölçeğin ve OHLC çizgisinin rengini alır.      |
//+------------------------------------------------------------------+
color ChartForeColorGet(const long chart_ID=0)
  {
//--- rengi almak için değişkeni hazırla
   long result=clrNONE;
//--- hata değerini sıfırla
   ResetLastError();
//--- eksenin, ölçeğin ve OHLC çizgisinin rengini al
   if(!ChartGetInteger(chart_ID,CHART_COLOR_FOREGROUND,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Fonksiyon; eksenin, ölçeğin ve OHLC çizgisinin rengini ayarlar.  |
//+------------------------------------------------------------------+
bool ChartForeColorSet(const color clr,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- eksenin, ölçeğin ve OHLC çizgisinin rengini ayarla
   if(!ChartSetInteger(chart_ID,CHART_COLOR_FOREGROUND,clr))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_COLOR_GRID** - çizelge ızgara rengi

```
//+------------------------------------------------------------------+
//| Fonksiyon, çizelge ızgara rengini alır.                          |
//+------------------------------------------------------------------+
color ChartGridColorGet(const long chart_ID=0)
  {
//--- rengi almak için değişkeni hazırla
   long result=clrNONE;
//--- hata değerini sıfırla
```

```
   ResetLastError();
//--- çizelge ızgara rengini al
   if(!ChartGetInteger(chart_ID,CHART_COLOR_GRID,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, çizelge ızgara rengini ayarlar.                       |
//+------------------------------------------------------------------+
bool ChartGridColorSet(const color clr,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- çizelge ızgara rengini ayarla
   if(!ChartSetInteger(chart_ID,CHART_COLOR_GRID,clr))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_COLOR_VOLUME** - hacimlerin ve pozisyon açma seviyelerinin rengini gösterir.

```
//+------------------------------------------------------------------+
//| Fonksiyon, hacimlerin ve piyasaya giriş seviyelerinin rengini    |
//| ayarlar.                                                         |
//+------------------------------------------------------------------+
color ChartVolumeColorGet(const long chart_ID=0)
  {
//--- rengi almak için değişkeni hazırla
   long result=clrNONE;
//--- hata değerini sıfırla
   ResetLastError();
//--- hacimlerin ve piyasaya giriş seviyelerinin rengini al
   if(!ChartGetInteger(chart_ID,CHART_COLOR_VOLUME,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((color)result);
```

```
  }
//+------------------------------------------------------------------+
//| Fonksiyon, hacimlerin ve piyasaya giriş seviyelerinin rengini    |
//| ayarlar.                                                         |
//+------------------------------------------------------------------+
bool ChartVolumeColorSet(const color clr,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- hacimlerin ve piyasaya giriş seviyelerinin rengini ayarla
   if(!ChartSetInteger(chart_ID,CHART_COLOR_VOLUME,clr))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_COLOR_CHART_UP** - yukarı yönlü çubuğun, gölgesinin ve alım yönlü mumun gövde sınırının rengi.

```
//+------------------------------------------------------------------+
//| Fonksiyon, yukarı yönlü çubuğun, gölgesinin ve alım yönlü mumun  |
//| gövde sınırının rengini alır                                     |
//+------------------------------------------------------------------+
color ChartUpColorGet(const long chart_ID=0)
  {
//--- rengi almak için değişkeni hazırla
   long result=clrNONE;
//--- hata değerini sıfırla
   ResetLastError();
//--- yukarı yönlü çubuğun, gölgesinin ve alım yönlü mumun gövde sınırının rengini al
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_UP,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, yukarı yönlü çubuğun, gölgesinin ve alım yönlü mumun  |
//| gövde sınırının rengini alır                                     |
//+------------------------------------------------------------------+
bool ChartUpColorSet(const color clr,const long chart_ID=0)
  {
```

```
//--- hata değerini sıfırla
   ResetLastError();
//--- yukarı yönlü çubuğun, gölgesinin ve alım yönlü mumun gövde sınırının rengini aya
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_UP,clr))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_COLOR_CHART_DOWN** - aşağı yönlü çubuğun, gölgesinin ve satım yönlü mumun gövde sınırının rengi.

```
//+------------------------------------------------------------------+
//| Fonksiyon, yukarı yönlü çubuğun, gölgesinin ve alım yönlü mumun  |
//| gövde sınırının rengini alır.                                    |
//+------------------------------------------------------------------+
color ChartDownColorGet(const long chart_ID=0)
  {
//--- rengi almak için değişkeni hazırla
   long result=clrNONE;
//--- hata değerini sıfırla
   ResetLastError();
//--- aşağı yönlü çubuğun, gölgesinin ve satım yönlü mumun gövde sınırının rengini al
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_DOWN,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, aşağı yönlü çubuğun, gölgesinin ve satım yönlü mumun  |
//| gövde sınırının rengini alır.                                    |
//+------------------------------------------------------------------+
bool ChartDownColorSet(const color clr,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- aşağı yönlü çubuğun, gölgesinin ve satım yönlü mumun gövde sınırının rengini aya
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_DOWN,clr))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
```

```
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_COLOR_CHART_LINE** - çizelge çizgisinin ve Doji mumlarının rengi.

```
//+------------------------------------------------------------------+
//| Fonksiyon, çizelge çizgisinin ve Doji mumlarının rengini alır.   |
//+------------------------------------------------------------------+
color ChartLineColorGet(const long chart_ID=0)
  {
//--- rengi almak için değişkeni hazırla
   long result=clrNONE;
//--- hata değerini sıfırla
   ResetLastError();
//--- çizelge çizgisinin ve Doji mumlarının rengini al
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CHART_LINE,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, çizelge çizgisinin ve Doji mumlarının rengini ayarlar |
//+------------------------------------------------------------------+
bool ChartLineColorSet(const color clr,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- çizelge çizgisinin ve Doji mumlarının rengini ayarla
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CHART_LINE,clr))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_COLOR_CANDLE_BULL** - alım yönlü mumların gövde rengi.

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

```
//| Fonksiyon, alım yönlü mumların gövde rengini alır.               |
//+------------------------------------------------------------------+
color ChartBullColorGet(const long chart_ID=0)
  {
//--- rengi almak için değişkeni hazırla
   long result=clrNONE;
//--- hata değerini sıfırla
   ResetLastError();
//--- alım yönlü mumların gövde rengini al
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, alım yönlü mumların gövde rengini ayarlar.            |
//+------------------------------------------------------------------+
bool ChartBullColorSet(const color clr,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- alım yönlü mumların gövde rengini ayarla
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BULL,clr))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_COLOR_CANDLE_BEAR** - satış yönlü mumların gövde rengi.

```
//+------------------------------------------------------------------+
//| Fonksiyon, satış yönlü mumların gövde rengini alır.              |
//+------------------------------------------------------------------+
color ChartBearColorGet(const long chart_ID=0)
  {
//--- rengi almak için değişkeni hazırla
   long result=clrNONE;
//--- hata değerini sıfırla
   ResetLastError();
//--- satış yönlü mumların gövde rengini al
   if(!ChartGetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,0,result))
```

```
      {
       //--- uzmanlar günlüğünde hata mesajını görüntüle
       Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      }
//--- çizelge özelliğinin değerini dön
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, satış yönlü mumların gövde rengini ayarlar.           |
//+------------------------------------------------------------------+
bool ChartBearColorSet(const color clr,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- satış yönlü mumların gövde rengini ayarla
   if(!ChartSetInteger(chart_ID,CHART_COLOR_CANDLE_BEAR,clr))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_COLOR_BID** - Satış fiyatı çizgisinin rengi.

```
//+------------------------------------------------------------------+
//| Fonksiyon, Satış çizgisinin rengini alır.                        |
//+------------------------------------------------------------------+
color ChartBidColorGet(const long chart_ID=0)
  {
//--- rengi almak için değişkeni hazırla
   long result=clrNONE;
//--- hata değerini sıfırla
   ResetLastError();
//--- Satış fiyatının çizgisinin rengini al
   if(!ChartGetInteger(chart_ID,CHART_COLOR_BID,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, Satış çizgisi rengini ayarlar.                        |
//+------------------------------------------------------------------+
```

```
bool ChartBidColorSet(const color clr,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- satış fiyatı çizgisi rengini ayarla
   if(!ChartSetInteger(chart_ID,CHART_COLOR_BID,clr))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_COLOR_ASK** - Alış fiyatı çizgisi rengi.

```
//+------------------------------------------------------------------+
//| Fonksiyon, Alış çizgisi rengini alır.                            |
//+------------------------------------------------------------------+
color ChartAskColorGet(const long chart_ID=0)
  {
//--- rengi almak için değişkeni hazırla
   long result=clrNONE;
//--- hata değerini sıfırla
   ResetLastError();
//--- alış fiyatı çizgisinin rengini al
   if(!ChartGetInteger(chart_ID,CHART_COLOR_ASK,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, Alış çizgisi rengini ayarlar.                         |
//+------------------------------------------------------------------+
bool ChartAskColorSet(const color clr,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- alış fiyatı çizgisinin rengini ayarla
   if(!ChartSetInteger(chart_ID,CHART_COLOR_ASK,clr))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
```

```
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_COLOR_LAST** - son gerçekleşen işlem fiyatı (Last) çizgisinin rengi.

```
//+------------------------------------------------------------------+
//| Fonksiyon, son gerçekleşen işlem fiyatı çizgisinin rengini alır. |
//+------------------------------------------------------------------+
color ChartLastColorGet(const long chart_ID=0)
  {
//--- rengi almak için değişkeni hazırla
   long result=clrNONE;
//--- hata değerini sıfırla
   ResetLastError();
//--- son gerçekleşen işlem fiyatı (Last) çizgisinin rengini al
   if(!ChartGetInteger(chart_ID,CHART_COLOR_LAST,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, son gerçekleşen işlem fiyatı çizgisinin rengini       |
//| ayarlar.                                                         |
//+------------------------------------------------------------------+
bool ChartLastColorSet(const color clr,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- son gerçekleşen işlem fiyatı (Last) çizgisinin rengini ayarla
   if(!ChartSetInteger(chart_ID,CHART_COLOR_LAST,clr))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_COLOR_STOP_LEVEL** - stop emri seviyesinin rengi (Stop Loss ve Take Profit).

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

```
//| Fonksiyon, Stop Loss ve Take Profit seviyelerinin rengini alır.    |
//+------------------------------------------------------------------+
color ChartStopLevelColorGet(const long chart_ID=0)
  {
//--- rengi almak için değişkeni hazırla
   long result=clrNONE;
//--- hata değerini sıfırla
   ResetLastError();
//--- stop (durdurma) emri seviyelerinin (Stop Loss ve Take Profit) rengini al
   if(!ChartGetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return((color)result);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, Stop Loss ve Take Profit seviyelerinin rengini ayarlar. |
//+------------------------------------------------------------------+
bool ChartStopLevelColorSet(const color clr,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- durdurma emri seviyelerinin (Stop Loss ve Take Profit) rengini ayarla
   if(!ChartSetInteger(chart_ID,CHART_COLOR_STOP_LEVEL,clr))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_SHOW_TRADE_LEVELS** - alım-satım seviyelerini çizelgede gösterme özelliği(Açık pozisyonların seviyeleri, Stop Loss, Take Profit ve bekleyen emirler).

```
//+------------------------------------------------------------------+
//| Fonksiyon, alım-satım seviyelerinin çizelgedeki gösterimini tanımlar |
//+------------------------------------------------------------------+
bool ChartShowTradeLevelsGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
```

```
   if(!ChartGetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------------------
//| Fonksiyon, alım-satım seviyelerinin gösterim modunu etkinleştirir/devre dışı bıra
//+------------------------------------------------------------------------------
bool ChartShowTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_SHOW_TRADE_LEVELS,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_DRAG_TRADE_LEVELS** - alım-satım seviyelerinin çizelge üstüne fare ile sürüklenmesi özelliği.

```
//+------------------------------------------------------------------------------+
//| Fonksiyon, alım-satım seviyelerinin çizelgede fare ile sürüklenmesine        |
//| izin veriliyor mu, tanımlar.                                                 |
//+------------------------------------------------------------------------------+
bool ChartDragTradeLevelsGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
```

```
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, alım-satım seviyelerinin çizelgede fare ile           |
//| sürüklenmesini etkinleştirir/devre dışı bırakır.                 |
//+------------------------------------------------------------------+
bool ChartDragTradeLevelsSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_DRAG_TRADE_LEVELS,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_SHOW_DATE_SCALE** - zaman ölçeğini çizelge üzerinde gösterme özelliği.

```
//+------------------------------------------------------------------+
//| Fonksiyon, zaman ölçeği çizelgede gösteriliyor mu, tanımlar.     |
//+------------------------------------------------------------------+
bool ChartShowDateScaleGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
```

```
//+------------------------------------------------------------------------------+
//| Fonksiyon, zaman ölçeğinin çizelgede gösterimini etkinleştirir veya          |
//| devre dışı bırakır.                                                          |
//+------------------------------------------------------------------------------+
bool ChartShowDateScaleSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_SHOW_DATE_SCALE,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_SHOW_PRICE_SCALE** - çizelgede fiyat ölçeğinin gösterilmesi özelliği.

```
//+------------------------------------------------------------------+
//| Fonksiyon, çizelgede fiyat ölçeği gösterilmiş mi, tanımlar.      |
//+------------------------------------------------------------------+
bool ChartShowPriceScaleGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, göstergede fiyat ölçeğinin gösterimini devreye sokar veya |
//| devre dışı bırakır.                                              |
//+------------------------------------------------------------------+
bool ChartShowPriceScaleSet(const bool value,const long chart_ID=0)
  {
```

```
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_SHOW_PRICE_SCALE,0,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_SHOW_ONE_CLICK** - property of displaying the "One click trading" panel on a chart.

```
//+------------------------------------------------------------------+
//| Checks if the "One click trading" panel is displayed on chart    |
//+------------------------------------------------------------------+
bool ChartShowOneClickPanelGet(bool &result,const long chart_ID=0)
  {
//--- özellik değerini almak için değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
     {
      //--- Uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini bellekte sakla
   result=value;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Enables/disables displaying of the "One click trading" panel     |
//| on chart                                                         |
//+------------------------------------------------------------------+
bool ChartShowOneClickPanelSet(const bool value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetInteger(chart_ID,CHART_SHOW_ONE_CLICK,0,value))
     {
      //--- Uzmanlar günlüğünde hata mesajını görüntüle
```

```
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_SHIFT_SIZE** - sıfır çubuğunun sağ sınırdan yüzdelik değerlerle kaydırılma boyutu.

```
//+------------------------------------------------------------------+
//| Fonksiyon, sıfır çubuğunun sağ sınırdan yüzdelik değerlerle |
//| (10%'dan 50%'ye kadar) kaydırılma boyutunu alır.             |
//+------------------------------------------------------------------+
double ChartShiftSizeGet(const long chart_ID=0)
  {
//--- sonucu almak için değişkeni hazırla
   double result=EMPTY_VALUE;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetDouble(chart_ID,CHART_SHIFT_SIZE,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return(result);
  }
//+------------------------------------------------------------------------+
//| Fonksiyon, sıfır çubuğunun sağ sınırdan yüzdelik değerlerle          |
//| (10%'dan 50%'ye kadar) kaydırılma boyutunu ayarlar. Kaydırma modunu |
//| devreye sokmak için, CHART_SHIFT özelliğinin değeri true olarak     |
//| ayarlanmalıdır.                                                      |
//+------------------------------------------------------------------------+
bool ChartShiftSizeSet(const double value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetDouble(chart_ID,CHART_SHIFT_SIZE,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_FIXED_POSITION** - sol sınırdan yüzdelik değerlerle çizelge sabitlenmiş konumu.

```
//+------------------------------------------------------------------+
//| Fonksiyon, sol sınırdan yüzdelik değerlerle çizelge sabitlenmiş |
//| konumunu ayarlar.                                                |
//+------------------------------------------------------------------+
double ChartFixedPositionGet(const long chart_ID=0)
  {
//--- sonucu almak için değişkeni hazırla
   double result=EMPTY_VALUE;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetDouble(chart_ID,CHART_FIXED_POSITION,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return(result);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, sol sınırdan yüzdelik değerlerle çizelge sabitlenmiş |
//| konumunu ayarlar.                Çizelge sabitlenmiş konumunun   |
//| yerini göstermek için, CHART_AUTOSCROLL özelliğinin              |
//| false şeklinde ayarlanmış olması gerekir.                        |
//+------------------------------------------------------------------+
bool ChartFixedPositionSet(const double value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetDouble(chart_ID,CHART_FIXED_POSITION,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_FIXED_MAX** - çizelgenin sabitlenmiş maksimum özelliği.

```
//+----------------------------------------------------------+
//| Fonksiyon, çizelgenin sabitlenmiş maksimum değerini alır |
```

```
//+------------------------------------------------------------------+
double ChartFixedMaxGet(const long chart_ID=0)
  {
//--- sonucu almak için değişkeni hazırla
   double result=EMPTY_VALUE;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MAX,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return(result);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, çizelgenin sabitlenmiş maksimum değerini ayarlar.     |
//| Özelliğin değerini değiştirmek için,                             |
//| CHART_SCALEFIX özelliğinin değeri öncelikle true şeklinde        |
//| ayarlanmalıdır.                                                  |
//+------------------------------------------------------------------+
bool ChartFixedMaxSet(const double value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MAX,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_FIXED_MIN** - çizelgenin sabitlenmiş minimum özelliği.

```
//+------------------------------------------------------------------+
//| Fonksiyon, çizelgenin sabitlenmiş minimum değerini alır.         |
//+------------------------------------------------------------------+
double ChartFixedMinGet(const long chart_ID=0)
  {
//--- sonucu almak için değişkeni hazırla
   double result=EMPTY_VALUE;
//--- hata değerini sıfırla
   ResetLastError();
```

```
//--- özellik değerini al
   if(!ChartGetDouble(chart_ID,CHART_FIXED_MIN,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return(result);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, çizelgenin sabitlenmiş minimum değerini ayarlar.      |
//| Özelliğin değerini değiştirmek için,                             |
//| CHART_SCALEFIX özelliğinin değeri öncelikle true şeklinde        |
//| ayarlanmalıdır.                                                  |
//+------------------------------------------------------------------+
bool ChartFixedMinSet(const double value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetDouble(chart_ID,CHART_FIXED_MIN,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_POINTS_PER_BAR** - çubuk başına düşen puan bazında ölçek değeri.

```
//+------------------------------------------------------------------------+
//| Fonksiyon, çubuk başına düşen puan bazında çizelge ölçek değerini alır.|
//+------------------------------------------------------------------------+
double ChartPointsPerBarGet(const long chart_ID=0)
  {
//--- sonucu almak için değişkeni hazırla
   double result=EMPTY_VALUE;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetDouble(chart_ID,CHART_POINTS_PER_BAR,0,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
```

```
   return(result);
  }
//+------------------------------------------------------------------------+
//| Fonksiyon, nokta bölü çubuk şeklindeki çizelge ölçek değerini ayarlar  |
//| Bu özelliğin değerinin değişmesinin sonucunu göstermek için,           |
//| CHART_SCALE_PT_PER_BAR özelliğinin değeri,                             |
//| öncelikli olarak true şeklinde ayarlanmalıdır.                         |
//+------------------------------------------------------------------------+
bool ChartPointsPerBarSet(const double value,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
   if(!ChartSetDouble(chart_ID,CHART_POINTS_PER_BAR,value))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_PRICE_MIN** çizelgenin minimum değerine dönüş yapar.

```
//+------------------------------------------------------------------------------------+
//| Fonksiyon, ana pencerenin veya alt pencerenin çizelge minimum değerini alır.       |
//|                                                                                    |
//+------------------------------------------------------------------------------------+
double ChartPriceMin(const long chart_ID=0,const int sub_window=0)
  {
//--- sonucu almak için değişkeni hazırla
   double result=EMPTY_VALUE;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MIN,sub_window,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return(result);
  }
```

- **CHART_PRICE_MAX** çizelgenin maksimum değerine dönüş yapar.

```
//+------------------------------------------------------------------------------+
//| Fonksiyon, ana pencerenin veya alt pencerenin çizelge maksimum değerini alır.  |
//|                                                                              |
//+------------------------------------------------------------------------------+
double ChartPriceMax(const long chart_ID=0,const int sub_window=0)
  {
//--- sonucu almak için değişkeni hazırla
   double result=EMPTY_VALUE;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetDouble(chart_ID,CHART_PRICE_MAX,sub_window,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
     }
//--- çizelge özelliğinin değerini dön
   return(result);
  }
```

- **CHART_COMMENT** - çizelge üstünde yorum.

```
//+------------------------------------------------------------------------+
//| Fonksiyon, çizelgenin sol üst köşesindeki yorumu alır.                 |
//+------------------------------------------------------------------------+
bool ChartCommentGet(string &result,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetString(chart_ID,CHART_COMMENT,result))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------------+
//| Fonksiyon, çizelgenin sol üst köşesindeki yorumu ayarlar.              |
//| devre dışı bırakır.                                                    |
//+------------------------------------------------------------------------+
bool ChartCommentSet(const string str,const long chart_ID=0)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini ayarla
```

```
   if(!ChartSetString(chart_ID,CHART_COMMENT,str))
     {
      //--- uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

- **CHART_IS_MAXIMIZED** - çizelge penceresi maksimize edildi.

```
//+------------------------------------------------------------------+
//| Mevcut çizelge penceresinin maksimize olup olmadığını denetler   |
//+------------------------------------------------------------------+
bool ChartWindowsIsMaximized(bool &result,const long chart_ID=0)
  {
//--- özellik değerinin kaydedileceği değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_IS_MAXIMIZED))
     {
      //--- Uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini değişkene ata
   result=value;
//--- işlem başarılı
   return(true);
  }
```

- **CHART_IS_MINIMIZED** - çizelge penceresi minimize edildi.

```
//+------------------------------------------------------------------+
//| Mevcut çizelge penceresinin minimize olup olmadığını denetler    |
//+------------------------------------------------------------------+
bool ChartWindowsIsMinimized(bool &result,const long chart_ID=0)
  {
//--- özellik değerinin kaydedileceği değişkeni hazırla
   long value;
//--- hata değerini sıfırla
   ResetLastError();
//--- özellik değerini al
   if(!ChartGetInteger(chart_ID,CHART_IS_MINIMIZED))
     {
      //--- Uzmanlar günlüğünde hata mesajını görüntüle
      Print(__FUNCTION__+", Hata Kodu = ",GetLastError());
      return(false);
     }
//--- çizelge özelliğinin değerini değişkene ata
   result=value;
//--- işlem başarılı
   return(true);
  }
```

Çizelge özellikleri için panel

```
//--- kontrol elemanlarının kütüphanesine bağlan
#include <ChartObjects\ChartObjectsTxtControls.mqh>
//--- ön tanımlı sabitler
#define X_PROPERTY_NAME_1    10  // ilk sütundaki özellik isminin x koordinatı
#define X_PROPERTY_VALUE_1   225 // ilk sütundaki özellik değerinin x koordinatı
#define X_PROPERTY_NAME_2    345 // ikinci ve üçüncü sütundaki özellik isminin x koord
#define X_PROPERTY_VALUE_2   550 // ikinci ve üçüncü sütundaki özellik değerinin x koo
#define X_BUTTON_1           285 // ilk sütundaki düğmenin x koordinatı
#define X_BUTTON_2           700 // ikinci ve üçüncü sütundaki düğmenin x koordinatı
#define Y_PROPERTY_1         30  // ilk ve ikinci sütunların başlangıç y koordinatları
#define Y_PROPERTY_2         286 // üçüncü sütunun başlangıç y koordinatı
#define Y_DISTANCE           16  // çizgiler arasındaki y eksensel uzaklıklar
#define LAST_PROPERTY_NUMBER 111 // son grafiksel özelliğin numarası
//--- giriş parametreleri
input color InpFirstColor=clrDodgerBlue; // Tek çizgilerin rengi
input color InpSecondColor=clrGoldenrod; // Çift çizgilerin rengi
//--- değişkenler ve diziler
CChartObjectLabel  ExtLabelsName[];  // özellik isimlerinin görüntülenmesi için etiket
CChartObjectLabel  ExtLabelsValue[]; // özellik değerlerinin görüntülenmesi için etike
CChartObjectButton ExtButtons[];     // düğmeler
int                ExtNumbers[];     // özellik indisleri
string             ExtNames[];       // özellik isimleri
uchar              ExtDataTypes[];   // özellik veri tipleri (tamsayı, double, string)
uint               ExtGroupTypes[];  // özelliklere ait veriyi gruplardan birine depol
uchar              ExtDrawTypes[];   // özellik görünümü tipine veri kaydeden dizi
double             ExtMaxValue[];    // panelle çalışırken gerçekleşen maksimum özelli
double             ExtMinValue[];    // panelle çalışırken gerçekleşen minimum özellil
```

```
double            ExtStep[];        // özelliklerin değiştirilmesi için adımlar
int               ExtCount;         // tüm özelliklerin toplam sayısı
color             ExtColors[2];     // çizgileri görüntülemek için gereken renklerin
string            ExtComments[2];   // yorumların dizisi (CHART_COMMENT özelliği için
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- çizelgede bir yorum göster
   Comment("SomeComment");
//--- renkleri, daha sonra aralarında değişim yapabilmek için diziye depola
   ExtColors[0]=InpFirstColor;
   ExtColors[1]=InpSecondColor;
//--- yorumları, daha sonra aralarında değişim yapabilmek için diziye depola
   ExtComments[0]="FirstComment";
   ExtComments[1]="SecondComment";
//--- çizelge özelliklerini yönetmek için, kontrol panelini hazırla ve göster
   if(!PrepareControls())
      return(INIT_FAILED);
//--- başarılı çalıştırma
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Deinitialization function of the expert                          |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- yorumu çizelgeden kaldır
   Comment("");
  }
//+------------------------------------------------------------------+
//| Handler of a chart event                                         |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- çizelge nesnesinin tıklanma olayını kontrol et
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      //--- nesne ismini ayraç ile böl
      string obj_name[];
      StringSplit(sparam,'_',obj_name);
      //--- nesne bir düğme mi, kontrol et
      if(obj_name[0]=="Button")
        {
         //--- düğmenin indisini al
```

```
         int index=(int)StringToInteger(obj_name[1]);
         //--- düğmeyi serbest bırak
         ExtButtons[index].State(false);
         //--- tipine bağlı olarak özelliğin yeni değerini ayarla
         if(ExtDataTypes[index]=='I')
            ChangeIntegerProperty(index);
         if(ExtDataTypes[index]=='D')
            ChangeDoubleProperty(index);
         if(ExtDataTypes[index]=='S')
            ChangeStringProperty(index);
        }
     }
//--- özellik değerlerini yeniden çiz
   RedrawProperties();
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| Çizelgenin tamsayı özelliğini değiştir                           |
//+------------------------------------------------------------------+
void ChangeIntegerProperty(const int index)
  {
//--- mevcut özellik değerini al
   long value=ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index]);
//--- sonraki özellik değerini ayarla
   switch(ExtDrawTypes[index])
     {
      case 'C':
         value=GetNextColor((color)value);
         break;
      default:
         value=(long)GetNextValue((double)value,index);
         break;
     }
//--- yeni özellik değerini ayarla
   ChartSetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[index],0,value);
  }
//+------------------------------------------------------------------+
//| Çizelgenin double özelliğini değiştir                            |
//+------------------------------------------------------------------+
void ChangeDoubleProperty(const int index)
  {
//--- mevcut özellik değerini al
   double value=ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index]);
//--- sonraki özellik değerini ayarla
   value=GetNextValue(value,index);
//--- yeni özellik değerini ayarla
   ChartSetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumbers[index],value);
  }
//+------------------------------------------------------------------+
```

```
//| Çizelgenin string özelliğini değiştir                            |
//+------------------------------------------------------------------+
void ChangeStringProperty(const int index)
  {
//--- ExtComments dizisi içinde değişim yapmak için statik değişken
   static uint comment_index=1;
//--- başka bir yorum almak için değişim indisi
   comment_index=1-comment_index;
//--- yeni özellik değerini ayarla
   ChartSetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[index],ExtComments[comment_
  }
//+------------------------------------------------------------------+
//| Sonraki özellik değerini tanımla                                 |
//+------------------------------------------------------------------+
double GetNextValue(const double value,const int index)
  {
   if(value+ExtStep[index]<=ExtMaxValue[index])
      return(value+ExtStep[index]);
   else
      return(ExtMinValue[index]);
  }
//+------------------------------------------------------------------+
//| color tipi özellik için sonraki rengi al                         |
//+------------------------------------------------------------------+
color GetNextColor(const color clr)
  {
//--- sonraki renk değerine dönüş yap
   switch(clr)
     {
      case clrWhite: return(clrRed);
      case clrRed:   return(clrGreen);
      case clrGreen: return(clrBlue);
      case clrBlue:  return(clrBlack);
      default:       return(clrWhite);
     }
  }
//+------------------------------------------------------------------+
//| Yeniden çizim özelliği değeri                                    |
//+------------------------------------------------------------------+
void RedrawProperties(void)
  {
//--- özellik değeri metni
   string text;
   long   value;
//--- özelliklerin sayısının döngüsü
   for(int i=0;i<ExtCount;i++)
     {
      text="";
      switch(ExtDataTypes[i])
```

```
         {
          case 'I':
             //--- mevcut özelliğin değerini al
             if(!ChartGetInteger(0,(ENUM_CHART_PROPERTY_INTEGER)ExtNumbers[i],0,value))
             break;
             //--- tamsayı özellik metni
             switch(ExtDrawTypes[i])
               {
                //--- color özelliği
                case 'C':
                   text=(string)((color)value);
                   break;
                   //--- mantıksal özellik
                case 'B':
                   text=(string)((bool)value);
                   break;
                   //--- ENUM_CHART_MODE sayımı özelliği
                case 'M':
                   text=EnumToString((ENUM_CHART_MODE)value);
                   break;
                   //--- ENUM_CHART_VOLUME_MODE sayımı özelliği
                case 'V':
                   text=EnumToString((ENUM_CHART_VOLUME_MODE)value);
                   break;
                   //--- int tipli sayı
                default:
                   text=IntegerToString(value);
                   break;
               }
             break;
          case 'D':
             //--- double özellik metni
             text=DoubleToString(ChartGetDouble(0,(ENUM_CHART_PROPERTY_DOUBLE)ExtNumber
             break;
          case 'S':
             //--- string özellik metni
             text=ChartGetString(0,(ENUM_CHART_PROPERTY_STRING)ExtNumbers[i]);
             break;
         }
       //--- özellik değerini görüntüle
       ExtLabelsValue[i].Description(text);
      }
   }
//+------------------------------------------------------------------+
//| Çizelge özelliklerini yönetmek için panel oluştur                |
//+------------------------------------------------------------------+
bool PrepareControls(void)
   {
//--- diziler için rezerv kullanarak bellek tahsis et
```

```
   MemoryAllocation(LAST_PROPERTY_NUMBER+1);
//--- değişkenler
   int i=0;      // döngü değişkeni
   int col_1=0; // ilk sütundaki özelliklerin sayısı
   int col_2=0; // ikinci sütundaki özelliklerin sayısın
   int col_3=0; // üçüncü sütundaki özelliklerin sayısı
//--- mevcut özellik sayısı - 0
   ExtCount=0;
//--- özellikleri döngüyle ara
   while(i<=LAST_PROPERTY_NUMBER)
     {
      //--- mevcut özellik sayısını muhafaza et
      ExtNumbers[ExtCount]=i;
      //--- döngü değişkeninin değerini artır
      i++;
      //--- bu numaraya sahip bir özellik var mı kontrol et
      if(CheckNumber(ExtNumbers[ExtCount],ExtNames[ExtCount],ExtDataTypes[ExtCount],Ex
        {
         //--- özellik için kontrol elemanları oluştur
         switch(ExtGroupTypes[ExtCount])
           {
            case 1:
               //--- özellik için etiketler ve bir düğme oluştur
               if(!ShowProperty(ExtCount,0,X_PROPERTY_NAME_1,X_PROPERTY_VALUE_1,X_BUTT
               return(false);
               //--- ilk sütundaki elemanların sayısı arttı
               col_1++;
               break;
            case 2:
               //--- özellik için etiketler ve bir düğme oluştur
               if(!ShowProperty(ExtCount,1,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,X_BUTT
               return(false);
               //--- ikinci sütundaki elemanların sayısı arttı
               col_2++;
               break;
            case 3:
               //--- özellik için sadece etiketler oluştur
               if(!ShowProperty(ExtCount,2,X_PROPERTY_NAME_2,X_PROPERTY_VALUE_2,0,Y_PR
               return(false);
               //--- üçüncü sütundaki elemanların sayısı arttı
               col_3++;
               break;
           }
         //--- maksimum ve minimum özellik değerini ve adımı tanımla
         GetMaxMinStep(ExtNumbers[ExtCount],ExtMaxValue[ExtCount],ExtMinValue[ExtCount
         //--- özellik değerini artır
         ExtCount++;
        }
     }
```

```
//--- diziler tarafından kullanılmayan belleği serbest bırak
   MemoryAllocation(ExtCount);
//--- özellik değerlerini yeniden çiz
   RedrawProperties();
   ChartRedraw();
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Diziler için bellek tahsis et                                    |
//+------------------------------------------------------------------+
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);
  }
//+------------------------------------------------------------------+
//| özellik indisinin, ENUM_CHART_PROPERTIES sayımlarından           |
//| biri olup olmadığını kontrol et                                  |
//+------------------------------------------------------------------+
bool CheckNumber(const int ind,string &name,uchar &data_type,uint &group_type,uchar &d
  {
//--- özellik tamsayı tipinde mi kontrol et
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_INTEGER)ind);
   if(_LastError==0)
     {
      data_type='I';                       // ENUM_CHART_PROPERTY_INTEGER sayımından öz
      GetTypes(ind,group_type,draw_type); // özellik görüntüleme parametrelerini tanır
      return(true);
     }
//--- özellik double tipli mi kontrol et
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_DOUBLE)ind);
   if(_LastError==0)
     {
      data_type='D';                       // ENUM_CHART_PROPERTY_DOUBLE sayımından öze
      GetTypes(ind,group_type,draw_type); // özellik görüntüleme parametrelerini tanır
      return(true);
     }
```

```
//--- özellik string tipli mi kontrol et
   ResetLastError();
   name=EnumToString((ENUM_CHART_PROPERTY_STRING)ind);
   if(_LastError==0)
     {
      data_type='S';                       // ENUM_CHART_PROPERTY_STRING sayımından öze
      GetTypes(ind,group_type,draw_type); // özellik görüntüleme parametrelerini tanır
      return(true);
     }
//--- özellik hiç bir sayıma ait değil
   return(false);
  }
//+------------------------------------------------------------------+
//| Özelliğin depolanacağı grubu ve aynı zamanda                     |
//| görüntülenme tipini tanımla                                      |
//+------------------------------------------------------------------+
void GetTypes(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- özellik üçüncü gruba mı ait, kontrol et
//--- üçüncü grup özellikleri üçüncü sütunda CHART_BRING_TO_TOP'dan başlayarak görünti
   if(CheckThirdGroup(property_number,group_type,draw_type))
      return;
//--- özellik ikinci gruba mı ait, kontrol et
//--- ikinci grup özellikleri ikinci sütunun başında gösterilir
   if(CheckSecondGroup(property_number,group_type,draw_type))
      return;
//--- eğer kendini burada bulduysan, özellik ilk gruba (ilk sütuna) aittir
   CheckFirstGroup(property_number,group_type,draw_type);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, özelliğin üçüncü gruba aitliğini kontrol eder ve      |
//| pozitif cevap durumunda onun görünüm tipini belirler             |
//+------------------------------------------------------------------+
bool CheckThirdGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- özellik üçüncü gruba mı ait, kontrol et
   switch(property_number)
     {
      //--- mantıksal özellikler
      case CHART_IS_OBJECT:
      case CHART_WINDOW_IS_VISIBLE:
         draw_type='B';
         break;
         //--- tamsayı özellikler
      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;
         //--- double özellikler
      case CHART_PRICE_MIN:
      case CHART_PRICE_MAX:
         draw_type='D';
         break;
         //--- gerçekte bu özellik, çizelge penceresinin diğer hepsinin üstünde göste
         //--- pencere diğerlerinin üstünde olduğu sürece, onu kullanmadan önce
         //--- panelin uygulanmasına gerek yoktur
      case CHART_BRING_TO_TOP:
         draw_type=' ';
         break;
         //--- özellik üçüncü gruba ait değil
      default:
         return(false);
     }
//--- özellik üçüncü gruba ait
   group_type=3;
   return(true);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, özelliğin ikinci gruba aitliğini kontrol eder         |
//| pozitif cevap durumunda onun görünüm tipini belirler             |
//+------------------------------------------------------------------+
bool CheckSecondGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- özellik ikinci gruba mı ait, kontrol et
   switch(property_number)
     {
      //--- ENUM_CHART_MODE tipi özellik
      case CHART_MODE:
         draw_type='M';
         break;
         //--- ENUM_CHART_VOLUME_MODE tipi özellik
      case CHART_SHOW_VOLUMES:
         draw_type='V';
         break;
         //--- dizgi özellik
      case CHART_COMMENT:
         draw_type='S';
         break;
         //--- renk özelliği
      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;
         //--- özellik ikinci gruba ait değil
      default:
         return(false);
     }
//--- özellik ikinci gruba ait
   group_type=2;
   return(true);
  }
//+------------------------------------------------------------------------+
//| Bu fonksiyon sadece, özelliğin ikinci ve üçüncü gruplara               |
//| ait olmadığı zaten bilinmekteyse çağrılır                              |
//+------------------------------------------------------------------------+
void CheckFirstGroup(const int property_number,uint &group_type,uchar &draw_type)
  {
//--- özellik ilk gruba ait
   group_type=1;
//--- özellik görüntüleme tipini tanımla
   switch(property_number)
     {
      //--- tamsayı özellikler
      case CHART_SCALE:
      case CHART_HEIGHT_IN_PIXELS:
         draw_type='I';
         return;
         //--- double özellikler
      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;
         //--- sadece mantıksal özellikler kaldı
      default:
         draw_type='B';
         return;
     }
  }
```

```
//+------------------------------------------------------------------+
//| Özellik için bir etiket ve bir düğme oluştur                     |
//+------------------------------------------------------------------+
bool ShowProperty(const int ind,const int type,const int x1,const int x2,
                  const int xb,const int y,const bool btn)
  {
//--- ExtColors renk dizisi içinde değişim için statik dizi
   static uint color_index[3]={1,1,1};
//--- başka bir renk almak için değişim indisi
   color_index[type]=1-color_index[type];
//--- (btn=true ise) özellik için etiketleri ve düğmeyi göster
   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);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Bir etiket oluştur                                               |
//+------------------------------------------------------------------+
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);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Düğmeyi oluştur                                                  |
//+------------------------------------------------------------------+
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);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
```

```
//| Maksimum ve minimum özellik değerini ve adımını tanımla          |
//+------------------------------------------------------------------+
void GetMaxMinStep(const int property_number,double &max,double &min,double &step)
  {
   double value;
//--- özellik tipine bağlı olarak değeri ayarla
   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;
         //--- varsayılan değerler
      default:
         max=1;
         min=0;
         step=1;
     }
  }
```

Nesne Sabitleri

Fiyat çizelgesinde oluşturulabilecek ve görüntülenebilecek 44 grafiksel nesne bulunmaktadır. Nesnelerle çalışmak için geliştirilmiş tüm sabitler 9 gruba bölünür:

- Nesne tipleri - grafiksel nesnelerin tanıtıcıları;
- Nesne özellikleri - grafiksel nesnelerin özelliklerinin alınması ve ayarlanması;
- Nesne tutturma yöntemleri - çizelgede nesnenin konumlandırılması için gereken sabitler;
- Tutturma köşesi - nesnenin üzerinde konumlandırılacağı çizelge köşesi;
- Nesnelerin görünürlüğü - bir nesnenin görünür olduğu zaman aralığının ayarlanması;
- Elliot Dalga Seviyeleri - dalga dereceleme işaretleri;
- Gann nesneleri - Gann yelpazesi ve Gann ızgarası için trend sabitleri;
- Web renkleri - ön tanımlı web renklerinin sabitleri;
- Wingdings - Wingdings yazı tipi karakterlerinin kodları.

Nesne Tipleri

ObjectCreate() fonksiyonu ile bir nesne oluşturuldığında, nesnenin tipinin belirlenmesi gerekir. Bu, ENUM_OBJECT sayımının değerlerinden biri olabilir. Grafiksel nesnelerle çalışmak amacıyla tasarlanmış fonksiyonları kullanarak, nesnelerin ileri özelliklerinin belirlenmesi mümkündür.

ENUM_OBJECT

Tanıtıcı		Açıklama
OBJ_VLINE		Dikey Çizgi
OBJ_HLINE		Yatay Çizgi
OBJ_TREND		Trend Çizgisi
OBJ_TRENDBYANGLE		Açılı Trend Çizgisi
OBJ_CYCLES		Döngü Çizgileri
OBJ_ARROWED_LINE		Oklu çizgi
OBJ_CHANNEL		Eşit-Aralıklı Kanal
OBJ_STDDEVCHANNEL		Standart Sapma Kanalı
OBJ_REGRESSION		Doğrusal Regresyon Kanalı
OBJ_PITCHFORK		Andrews Çatalı (Üçlü Çizgi)
OBJ_GANNLINE		Gann Çizgisi
OBJ_GANNFAN		Gann Yelpazesi
OBJ_GANNGRID		Gann Izgarası
OBJ_FIBO		Fibonacci Düzeltme Seviyeleri
OBJ_FIBOTIMES		Fibonacci Zaman Dilimleri
OBJ_FIBOFAN		Fibonacci Yelpazesi
OBJ_FIBOARC		Fibonacci Yayları
OBJ_FIBOCHANNEL		Fibonacci Kanalı
OBJ_EXPANSION		Fibonacci Açılımı
OBJ_ELLIOTWAVE5		Elliott Dürtü Dalgası
OBJ_ELLIOTWAVE3		Elliott Düzeltme Dalgası
OBJ_RECTANGLE		Dikdörtgen
OBJ_TRIANGLE		Üçgen
OBJ_ELLIPSE		Elips
OBJ_ARROW_THUMB_UP		İyi (Başparmak Yukarı)
OBJ_ARROW_THUMB_DOWN		Kötü (Başparmak Aşağı)

Tanıtıcı		Açıklama
OBJ_ARROW_UP		Yukarı Ok
OBJ_ARROW_DOWN		Aşağı Ok
OBJ_ARROW_STOP		Dur (Stop) İşareti
OBJ_ARROW_CHECK		Kontrol İşareti
OBJ_ARROW_LEFT_PRICE		Sol Fiyat Etiketi
OBJ_ARROW_RIGHT_PRICE		Sağ Fiyat Etiketi
OBJ_ARROW_BUY		Alış Sinyali
OBJ_ARROW_SELL		Satış Sinyali
OBJ_ARROW		Ok
OBJ_TEXT		Metin
OBJ_LABEL		Etiket
OBJ_BUTTON		Düğme
OBJ_CHART		Çizelge
OBJ_BITMAP		Bitmap Etiketi
OBJ_BITMAP_LABEL		Bitmap Etiketi
OBJ_EDIT		Düzenle
OBJ_EVENT		Ekonomik takvimdeki bir olaya karşılık gelen "Olay" nesnesi
OBJ_RECTANGLE_LABEL		Özel grafiksel arayüzler oluşturmak ve tasarlamak için "Dikdörtgen Etiket" nesnesi.

OBJ_VLINE

Diey çizgi.

Not

Bir dikey çizgi çizerken, çizginin görüntülenme modunu, tüm çizelge pencereleri için ayarlamak mümkündür (OBJPROP_RAY özelliği).

Örnek

Aşağıdaki script, çizelge üzerinde bir dikey çizgi oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Dikey Çizgi\" grafiksel nesnesini çizer."
#property description "Tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="VLine";      // Çizgi ismi
input int             InpDate=25;           // Olay tarihi, %
input color           InpColor=clrRed;      // Çizgi rengi
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Çizgi stili
input int             InpWidth=3;           // Çizgi genişliği
input bool            InpBack=false;        // Arkaplan nesnesi
input bool            InpSelection=true;    // Taşımak için vurgula
input bool            InpRay=true;          // Çizginin aşağı doğru sürekliliği
```

```
input bool            InpHidden=true;      // Nesne listesinde gizle
input long            InpZOrder=0;         // Fare tıklaması önceliği
//+------------------------------------------------------------------+
//| Dikey çizgiyi oluştur                                            |
//+------------------------------------------------------------------+
bool VLineCreate(const long            chart_ID=0,        // çizelge tanımlayıcısı
                 const string          name="VLine",      // çizgi ismi
                 const int             sub_window=0,      // alt pencere indisi
                 datetime              time=0,            // çizgi zamanı
                 const color           clr=clrRed,        // çizgi rengi
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // çizgi stili
                 const int             width=1,           // çizgi genişliği
                 const bool            back=false,        // arkaplan nesnesi
                 const bool            selection=true,    // taşıma için vurgula
                 const bool            ray=true,          // çizginin aşağı doğru süre
                 const bool            hidden=true,       // nesne listesinde gizle
                 const long            z_order=0)         // fare ttıklaması önceliği
  {
//--- çizgi zamanı ayarlanmamışsa, son çubuk üzerinde çiz
   if(!time)
      time=TimeCurrent();
//--- hata değerini sıfırla
   ResetLastError();
//--- bir dikey çizgi oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_VLINE,sub_window,time,0))
     {
      Print(__FUNCTION__,
            ": dikey çizginin oluşturulması başarısız oldu! Hata kodu = ",GetLastErro
      return(false);
     }
//--- çizgi rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- çizgi görünüm stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- çizgi genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- çizgiyi fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- çizgiyi, çizelge alt pencerelerinde görüntüleme modunu etkinleştir (true) veya
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY,ray);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Dikey çizgiyi taşı                                               |
//+------------------------------------------------------------------+
bool VLineMove(const long   chart_ID=0,   // çizelge kimliği
               const string name="VLine", // çizgi ismi
               datetime     time=0)       // çizgi zamanı
  {
//--- eğer, çizginin zamanı ayarlanmamışsa, çizgiyi son çubuğa taşı
   if(!time)
      time=TimeCurrent();
//--- hata değerini sıfırla
   ResetLastError();
//--- dikey çizgiyi taşı
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            ": dikey çizginin taşınması başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Dikey çizgiyi sil                                                |
//+------------------------------------------------------------------+
bool VLineDelete(const long   chart_ID=0,   // çizelge kimliği
                 const string name="VLine") // çizgi ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- dikey çizgiyi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": dikey çizginin silinmesi başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

```
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate<0 || InpDate>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- çizginin tutturma noktası koordinatlarını ayarlamak ve değiştirmek için
//--- tarih değerlerini depolayacak bir dizi
   datetime date[];
//--- bellek tahsisi
   ArrayResize(date,bars);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- çizgiyi çizmek için noktaları tanımla
   int d=InpDate*(bars-1)/100;
//--- bir dikey çizgi oluştur
   if(!VLineCreate(0,InpName,0,date[d],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpRay,InpHidden,InpZOrder))
      return;
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, çizgiyi taşı
//--- döngü sayacı
   int h_steps=bars/2;
//--- çizgiyi taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d<bars-1)
         d+=1;
      //--- tutturma noktasını taşı
      if(!VLineMove(0,InpName,date[d]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.03 saniyelik gecikme
      Sleep(30);
     }
```

```
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- kanalı çizelgeden sil
   VLineDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_HLINE

Yatay çizgi.

Örnek

Aşağıdaki script, bir yatay çizgi oluşturur ve onu çizelge üzerinde taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Horizontal Line\" grafiksel nesnesini çizer."
#property description "Tutturma (çapa) noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="HLine";      // Çizgi ismi
input int             InpPrice=25;          // Çizgi fiyatı, %
input color           InpColor=clrRed;      // Çizgi rengi
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Çizgi stili
input int             InpWidth=3;           // Çizgi genişliği
input bool            InpBack=false;        // Arkaplan nesnesi
input bool            InpSelection=true;    // Taşımak için vurgula
input bool            InpHidden=true;       // Nesne listesinde gizle
input long            InpZOrder=0;          // Fare tıklaması önceliği
//+------------------------------------------------------------------+
//| Yatay çizgiyi oluştur                                            |
//+------------------------------------------------------------------+
```

```
bool HLineCreate(const long            chart_ID=0,        // çizelge kimliği
                 const string          name="HLine",      // çizelge ismi
                 const int             sub_window=0,      // alt pencere indisi
                 double                price=0,           // çizgi fiyatı
                 const color           clr=clrRed,        // çizgi rengi
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // çizgi stili
                 const int             width=1,           // çizgi genişliği
                 const bool            back=false,        // arkaplan nesnesi
                 const bool            selection=true,    // taşıma için vurgula
                 const bool            hidden=true,       // nesne listesinde gizle
                 const long            z_order=0)         // fare ttıklaması önceliği
  {
//--- eğer fiyat ayarlanmamışsa, mevcut Bid fiyatı seviyesine ayarla
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- yatay çizgiyi oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_HLINE,sub_window,0,price))
     {
      Print(__FUNCTION__,
            ": yatay çizgi oluşturulamadı! Hata kodu = ",GetLastError());
      return(false);
     }
//--- çizgi rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- çizgi görünüm stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- çizgi genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- çizgiyi fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Yatay çizgiyi taşı                                               |
//+------------------------------------------------------------------+
bool HLineMove(const long   chart_ID=0,   // çizelge kimliği
```

```
                const string name="HLine", // çizgi ismi
                double       price=0)      // çizgi fiyatı
  {
//--- eğer fiyat ayarlanmamışsa, mevcut Bid fiyatı seviyesine ayarla
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- yatay çizgiyi taşı
   if(!ObjectMove(chart_ID,name,0,0,price))
     {
      Print(__FUNCTION__,
            ": yatay çizginin taşınması başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Yatay çizgiyi sil                                                |
//+------------------------------------------------------------------+
bool HLineDelete(const long   chart_ID=0,   // çizelge kimliği
                 const string name="HLine") // çizgi ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- yatay çizgiyi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": yatay çizginin silinmesi başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpPrice<0 || InpPrice>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
```

```
//--- tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih değerlerini depolayacak bir dizi
   double price[];
//--- bellek tahsisi
   ArrayResize(price,accuracy);
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- çizgiyi çizmek için noktaları tanımla
   int p=InpPrice*(accuracy-1)/100;
//--- yatay çizgiyi oluştur
   if(!HLineCreate(0,InpName,0,price[p],InpColor,InpStyle,InpWidth,InpBack,
      InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, çizgiyi taşı
//--- döngü sayacı
   int v_steps=accuracy/2;
//--- çizgiyi taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p<accuracy-1)
         p+=1;
      //--- tutturma noktasını taşı
      if(!HLineMove(0,InpName,price[p]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- çizelgeden sil
   HLineDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
```

```
//---
  }
```

OBJ_TREND

Trend Çizgisi.

Not

Trend Çizgisinin, sağa ve/veya sola doğru olan sürekliliklerinin (sırasıyla OBJPROP_RAY_RIGHT ve OBJPROP_RAY_LEFT özelliklerinin) belirtilmesi mümkündür.

Örnek

Aşağıdaki script Trend Çizgisini çizelge üzerinde oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Trend Çizgisi\" grafiksel nesnesini çizer."
#property description "Tutturma (çapa) noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="Trend";      // Çizgi ismi
input int             InpDate1=35;          // 1-inci noktanın tarihi, %
input int             InpPrice1=60;         // 1-inci noktanın fiyatı, %
input int             InpDate2=65;          // 2-inci noktanın tarihi, %
input int             InpPrice2=40;         // 2-inci noktanın fiyatı, %
input color           InpColor=clrRed;      // Çizgi rengi
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Çizgi stili
```

```
input int              InpWidth=2;          // Çizgi genişliği
input bool             InpBack=false;       // Arkaplan nesnesi
input bool             InpSelection=true;   // Taşımak için vurgula
input bool             InpRayLeft=false;    // Çizginin sola doğru sürekliliği
input bool             InpRayRight=false;   // Çizginin sağa doğru sürekliliği
input bool             InpHidden=true;      // Nesne listesinde gizle
input long             InpZOrder=0;         // Fare tıklaması önceliği
//+------------------------------------------------------------------+
//| Verilen koordinatlar ile bir trend çizgisi oluştur               |
//+------------------------------------------------------------------+
bool TrendCreate(const long            chart_ID=0,        // çizelge kimliği
                 const string          name="TrendLine",  // çizelge ismi
                 const int             sub_window=0,      // alt pencere indisi
                 datetime              time1=0,           // ilk noktanın zamanı
                 double                price1=0,          // ilk noktanın fiyatı
                 datetime              time2=0,           // ikinci noktanın zamanı
                 double                price2=0,          // ikinci noktanın fiyatı
                 const color           clr=clrRed,        // çizgi rengi
                 const ENUM_LINE_STYLE style=STYLE_SOLID, // çizgi stili
                 const int             width=1,           // çizgi genişliği
                 const bool            back=false,        // arkaplan nesnesi
                 const bool            selection=true,    // taşıma için vurgula
                 const bool            ray_left=false,    // çizginin sola doğru sürel
                 const bool            ray_right=false,   // çizginin sağa doğru sürel
                 const bool            hidden=true,       // nesne listesinde gizle
                 const long            z_order=0)         // fare ttıklaması önceliği
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeTrendEmptyPoints(time1,price1,time2,price2);
//--- hata değerini sıfırla
   ResetLastError();
//--- verilen koordinatlar ile trend çizgisini oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_TREND,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": trend çizgisinin oluşturulması başarısız oldu! Hata kodu = ",GetLastEr
      return(false);
     }
//--- çizgi rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- çizgi görünüm stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- çizgi genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- çizgiyi fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
```

```
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- çizginin sola doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- çizginin sağa doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Trend çizgisinin tutturma noktasını taşı                         |
//+------------------------------------------------------------------+
bool TrendPointChange(const long   chart_ID=0,       // çizelge kimliği
                      const string name="TrendLine", // çizgi ismi
                      const int    point_index=0,    // tutturma noktası indisi
                      datetime     time=0,           // tutturma noktasının zaman koo
                      double       price=0)          // tutturma noktasının zaman koo
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğ
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- trend çizgisinin tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, trend çizgisini çizelgeden siler.                     |
//+------------------------------------------------------------------+
bool TrendDelete(const long   chart_ID=0,       // çizelge kimliği
                 const string name="TrendLine") // çizgi ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- trend çizgisini sil
```

```
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": trend çizgisi silinemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Trend çizgisinin tutturma noktası değerlerini kontrol et ve      |
//| boş olanlar için varsayılan değerleri kullan                     |
//+------------------------------------------------------------------+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- ilk noktanın zamanı ayarlanmamışsa mevcut çubuk üzerinde olacak
   if(!time1)
      time1=TimeCurrent();
//--- ilk noktanın fiyatı ayarlanmamışsa mevcut Bid değerini alacak
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- ikinci noktanın zamanı ayarlanmamışsa, ilkinin 9 çubuk sonrasına konumlandırılır
   if(!time2)
     {
      //--- son 10 çubuğun açılış zamanını alacak bir dizi
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- ikinci noktayı birincisinden 9 çubuk ileri ayarla
      time2=temp[0];
     }
//--- ikinci noktanın fiyatı ayarlanmamışsa birinciye eşit olur
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

```
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- çizgiyi çizmek için noktaları tanımla
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- bir trend çizgisi oluştur
   if(!TrendCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,InpStyle
      InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, çizginin tutturma noktasını taşı
//--- döngü sayacı
   int v_steps=accuracy/5;
//--- ilk tutturma noktasını dikey olarak taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p1>1)
         p1-=1;
      //--- tutturma noktasını taşı
      if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
```

```
         return;
       //--- script işlemi devre dışı bırakıldı mı kontrol et
       if(IsStopped())
         return;
       //--- çizelgeyi yenile
       ChartRedraw();
      }
//--- ikinci tutturma noktasını dikey olarak taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p2<accuracy-1)
         p2+=1;
      //--- tutturma noktasını taşı
      if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- yarım saniyelik gecikme
   Sleep(500);
//--- döngü sayacı
   int h_steps=bars/2;
//--- iki tutturma noktasını aynı anda taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- şu değerleri kullan
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- noktaları kaydır
      if(!TrendPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!TrendPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.03 saniyelik gecikme
      Sleep(30);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
```

```
//--- trend çizgisini sil
   TrendDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_TRENDBYANGLE

Açılı Trend Çizgisi.

Not

Açılı Trend Çizgisinin, sağa ve/veya sola doğru olan sürekliliklerinin (sırasıyla OBJPROP_RAY_RIGHT ve OBJPROP_RAY_LEFT özelliklerinin) belirtilmesi mümkündür.

Açının ve ikinci tutturma noktasının koordinatlarının her ikisi de çizgi eğimini ayarlamak için kullanılabilir.

Örnek

Aşağıdaki script Trend Çizgisini çizelge üzerinde oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script \"Açılı Trend Çizgisi\" grafiksel nesnesini çizer."
#property description "Tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="Trend";      // Çizgi ismi
input int             InpDate1=50;          // 1-inci noktanın tarihi, %
input int             InpPrice1=75;         // 1-inci noktanın fiyatı, %
input int             InpAngle=0;           // Çizginin eğim açısı
input color           InpColor=clrRed;      // Çizgi rengi
```

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASH; // Çizgi stili
input int             InpWidth=2;          // Çizgi genişliği
input bool            InpBack=false;       // Arkaplan nesnesi
input bool            InpSelection=true;   // Taşımak için vurgula
input bool            InpRayLeft=false;    // Çizginin sola doğru sürekliliği
input bool            InpRayRight=true;    // Çizginin sağa doğru sürekliliği
input bool            InpHidden=true;      // Nesne listesinde gizle
input long            InpZOrder=0;         // Fare tıklaması önceliği
//+------------------------------------------------------------------+
//| Açı kullanarak trend çizgisi oluştur                             |
//+------------------------------------------------------------------+
bool TrendByAngleCreate(const long            chart_ID=0,        // çizelge kimliği
                        const string          name="TrendLine",  // çizgi ismi/t7>
                        const int             sub_window=0,      // alt-pencere numara
                        datetime              time=0,            // noktanın zamanı
                        double                price=0,           // noktanın fiyatı
                        const double          angle=45.0,        // eğim açısı
                        const color           clr=clrRed,        // çizgi rengi
                        const ENUM_LINE_STYLE style=STYLE_SOLID, // çizi stili
                        const int             width=1,           // çizgi genişliği
                        const bool            back=false,        // arka-planda
                        const bool            selection=true,    // taşıma için vurgul
                        const bool            ray_left=false,    // çizginin sola doğ
                        const bool            ray_right=true,    // çizginin sağa doğ
                        const bool            hidden=true,       // nesne listesinde
                        const long            z_order=0)         // fare tıklaması iç
  {
//--- trend çizgisinin fare ile taşınmasına imkan sağlamak için ikinci noktayı oluştu
   datetime time2=0;
   double   price2=0;
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeTrendEmptyPoints(time,price,time2,price2);
//--- hata değerini sıfırla
   ResetLastError();
//--- 2 nokta kullanarak bir trend çizgisi oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_TRENDBYANGLE,sub_window,time,price,time2,price2)
     {
      Print(__FUNCTION__,
            ": trend çizgisinin oluşturulması başarısız oldu! Hata kodu = ",GetLastEr
      return(false);
     }
//--- trend çizgisinin eğim açısını değiştir; açıyı değiştirirken, çizginin ikinci no
//--- koordinatları, açının yeni değerine göre otomatik olarak yeniden tanımlanır
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- çizgi rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- çizgi genişliğini ayarla
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- çizgiyi fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- çizginin sola doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- çizginin sağa doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Trend çizgisinin tutturma noktası koordinatlarını değiştir       |
//+------------------------------------------------------------------+
bool TrendPointChange(const long   chart_ID=0,       // çizelge kimliği
                      const string name="TrendLine", // çizgi ismi
                      datetime     time=0,           // tutturma noktasının zaman koo
                      double       price=0)          // tutturma noktasının zaman koo
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğ
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- trend çizgisinin tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Trend çizgisinin eğim açısını değiştir                           |
//+------------------------------------------------------------------+
bool TrendAngleChange(const long   chart_ID=0,       // çizelge kimliği
```

```
                      const string name="TrendLine", // trend çizgisi ismi
                      const double angle=45)         // trend çizgisi eğim açısı
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- trend çizgisinin eğim açısını değiştir
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            ": çizginin eğim açısı değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Trend çizgisini sil                                              |
//+------------------------------------------------------------------+
bool TrendDelete(const long   chart_ID=0,       // çizelge kimliği
                 const string name="TrendLine") // çizgi ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- trend çizgisini sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": trend çizgisi silinemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Trend çizgisinin tutturma noktası değerlerini kontrol et ve      |
//| boş olanlar için varsayılan değerleri kullan                     |
//+------------------------------------------------------------------+
void ChangeTrendEmptyPoints(datetime &time1,double &price1,
                            datetime &time2,double &price2)
  {
//--- ilk noktanın zamanı ayarlanmamışsa mevcut çubuk üzerinde olacak
   if(!time1)
      time1=TimeCurrent();
//--- ilk noktanın fiyatı ayarlanmamışsa mevcut Bid değerini alacak
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- ikinci, yardımcı nokta için koordinatları ayarla,
//--- ikinci nokta 9 çubuk solda ve aynı fiyatta olacak
   datetime second_point_time[10];
```

```
   CopyTime(Symbol(),Period(),time1,10,second_point_time);
   time2=second_point_time[0];
   price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- çizgiyi çizmek için noktaları tanımla
   int d1=InpDate1*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- bir trend çizgisi oluştur
   if(!TrendByAngleCreate(0,InpName,0,date[d1],price[p1],InpAngle,InpColor,InpStyle,
      InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
```

```
      }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, çizgiyi taşı
//--- döngü sayacı
   int v_steps=accuracy/2;
//--- tutturma noktasını taşı ve çizginin eğim açısını değiştir
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p1>1)
         p1-=1;
      //--- tutturma noktasını taşı
      if(!TrendPointChange(0,InpName,date[d1],price[p1]))
         return;
      if(!TrendAngleChange(0,InpName,18*(i+1)))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- çizelgeden sil
   TrendDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_CYCLES

Döngü Çizgileri.

Not

Çizgiler arasındaki uzaklık, nesnenin iki tutturma noktasının zaman koordinatları ile belirlenir.

Örnek

Aşağıdaki script, çizelge üzerinde döngü çizgilerini oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, çizelge üzerinde döngü çizgilerini oluşturur."
#property description "tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="Cycles";   // Nesne ismi
input int             InpDate1=10;        // 1-inci noktanın tarihi, %
input int             InpPrice1=45;       // 1-inci noktanın fiyatı, %
input int             InpDate2=20;        // 2-inci noktanın tarihi, %
input int             InpPrice2=55;       // 2-inci noktanın fiyatı, %
input color           InpColor=clrRed;    // Döngü çizgilerinin rengi
input ENUM_LINE_STYLE InpStyle=STYLE_DOT; // Döngü çizgilerinin stili
input int             InpWidth=1;         // Döngü çizgilerinin genişliği
```

```
input bool             InpBack=false;       // Arkaplan nesnesi
input bool             InpSelection=true;   // Taşıma için vurgula
input bool             InpHidden=true;      // Nesne listesinde gizle
input long             InpZOrder=0;         // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Döngü çizgilerini oluştur                                        |
//+------------------------------------------------------------------+
bool CyclesCreate(const long            chart_ID=0,         // çizelge kimliği
                  const string          name="Cycles",      // nesne ismi
                  const int             sub_window=0,       // alt-pencere indisi
                  datetime              time1=0,            // ilk nokta zamanı
                  double                price1=0,           // ilk nokta fiyatı
                  datetime              time2=0,            // ikinci nokta zamanı
                  double                price2=0,           // ikinci nokta fiyatı
                  const color           clr=clrRed,         // döngü çizgilerinin rengi
                  const ENUM_LINE_STYLE style=STYLE_SOLID,  // döngü çizgilerinin stili
                  const int             width=1,            // döngü çizgilerinin genişl
                  const bool            back=false,         // arka-planda
                  const bool            selection=true,     // taşıma için vurgula
                  const bool            hidden=true,        // nesne listesinde gizle
                  const long            z_order=0)          // fare tıklama önceliği
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeCyclesEmptyPoints(time1,price1,time2,price2);
//--- hata değerini sıfırla
   ResetLastError();
//--- verilen koordinatlarda döngü çizgilerini oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_CYCLES,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": döngü çizgilerinin oluşturulması başarısız oldu! Hata kodu = ",GetLastE
      return(false);
     }
//--- çizgilerin rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- çizgilerin görünüm stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- çizgilerin genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- çizgileri fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

```
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktasını taşı                                          |
//+------------------------------------------------------------------+
bool CyclesPointChange(const long   chart_ID=0,    // çizelge kimliği
                       const string name="Cycles", // nesne ismi
                       const int    point_index=0, // tutturma noktası indisi
                       datetime     time=0,        // tutturma noktası zaman koordinat
                       double       price=0)       // tutturma noktası fiyat koordinat
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Döngü çizgilerini sil                                            |
//+------------------------------------------------------------------+
bool CyclesDelete(const long   chart_ID=0,    // çizelge kimliği
                  const string name="Cycles") // nesne ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- döngü çizgilerini sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": döngü çizgilerinin silinmesi başarısız oldu! Hata kodu = ",GetLastErro
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

```
//+------------------------------------------------------------------+
//| döngü çizgilerinin tutturma noktası değerlerini kontrol et ve    |
//| boş olanlar için varsayılan değerleri ayarla                     |
//+------------------------------------------------------------------+
void ChangeCyclesEmptyPoints(datetime &time1,double &price1,
                             datetime &time2,double &price2)
  {
//--- ilk noktanın zamanı ayarlanmamışsa mevcut çubuk üzerinde olacak
   if(!time1)
      time1=TimeCurrent();
//--- ilk noktanın fiyatı ayarlanmamışsa mevcut Bid değerini alacak
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- ikinci noktanın zamanı ayarlanmamışsa, ilkinin 9 çubuk sonrasına konumlandırılı
   if(!time2)
     {
      //--- son 10 çubuğun açılış zamanını alacak bir dizi
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- ikinci noktayı birincisinden 9 çubuk ileri ayarla
      time2=temp[0];
     }
//--- ikinci noktanın fiyatı ayarlanmamışsa birinciye eşit olur
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- döngü çizgilerinin tutturma noktalarını ayarlamak ve değiştirmek için
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
```

```
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- döngü çizgilerinin çizileceği noktaları tanımla
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- bir trend çizgisi oluştur
   if(!CyclesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, tutturma noktalarını taşı
//--- döngü sayacı
   int h_steps=bars/5;
//--- ikinci tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d2<bars-1)
         d2+=1;
      //--- tutturma noktasını taşı
      if(!CyclesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
```

```
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   h_steps=bars/4;
//--- ilk tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d1<bars-1)
         d1+=1;
      //--- tutturma noktasını taşı
      if(!CyclesPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- nesneyi çizelgeden sil
   CyclesDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_ARROWED_LINE

Oklu çizgi.

Örnek

Aşağıdaki script, bir oklu çizgi oluşturur ve onu çizelge üzerinde taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Oklu çizgi\" grafiksel nesnesini çizer."
#property description "Tutturma (çapa) noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="ArrowedLine"; // Çizgi ismi
input int             InpDate1=35;           // 1-inci noktanın tarihi, %
input int             InpPrice1=60;          // 1-inci noktanın fiyatı, %
input int             InpDate2=65;           // 2-inci noktanın tarihi, %
input int             InpPrice2=40;          // 1-inci noktanın fiyatı, %
input color           InpColor=clrRed;       // Çizgi rengi
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;   // Çizgi stili
input int             InpWidth=2;            // Çizgi genişliği
input bool            InpBack=false;         // Arka-plan çizgisi
input bool            InpSelection=true;     // Taşıma için vurgula
input bool            InpHidden=true;        // Nesne listesinde gizle
input long            InpZOrder=0;           // Fare tıklama önceliği
```

```
//+------------------------------------------------------------------+
//| Verilen koordinatlarda bir oklu çizgi oluştur                    |
//+------------------------------------------------------------------+
bool ArrowedLineCreate(const long            chart_ID=0,         // çizelge kimliği
                       const string          name="ArrowedLine", // çizgi ismi
                       const int             sub_window=0,       // alt-pencere indisi
                       datetime              time1=0,            // ilk noktanın zaman
                       double                price1=0,           // ilk noktanın fiyat
                       datetime              time2=0,            // ikinci noktanın za
                       double                price2=0,           // ikinci noktanın fi
                       const color           clr=clrRed,         // çizgi rengi
                       const ENUM_LINE_STYLE style=STYLE_SOLID,  // çizgi stili
                       const int             width=1,            // çizgi genişliği
                       const bool            back=false,         // arka-planda
                       const bool            selection=true,     // taşımak için vurgu
                       const bool            hidden=true,        // nesne listesinde s
                       const long            z_order=0)          // fare tıklaması içi
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeArrowedLineEmptyPoints(time1,price1,time2,price2);
//--- hata değerini sıfırla
   ResetLastError();
//--- verilen koordinatlarla bir oklu çizgi oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_ARROWED_LINE,sub_window,time1,price1,time2,price
     {
      Print(__FUNCTION__,
            ": oklu çizgi oluşturulması başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- çizgi rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- çizgi görünüm stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- çizgi genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- çizgiyi fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
```

```
  }
//+------------------------------------------------------------------+
//| Oklu çizginin çapa noktasını taşı                                |
//+------------------------------------------------------------------+
bool ArrowedLinePointChange(const long   chart_ID=0,          // çizelge kimliği
                            const string name="ArrowedLine", // çizgi ismi
                            const int    point_index=0,      // tutturma (çapa) noktas
                            datetime     time=0,             // tutturma noktası zaman
                            double       price=0)            // tutturma noktası fiyat
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- çizginin tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, oklu çizgiyi çizelgeden siler                         |
//+------------------------------------------------------------------+
bool ArrowedLineDelete(const long   chart_ID=0,         // çizelge tanımlayıcısı
                       const string name="ArrowedLine") // çizgi ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- bir oklu çizgiyi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": oklu çizgi oluşturulması başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| tutturma noktalarını kontrol et ve                               |
//| boş olanlar için varsayılan değerleri ayarla                     |
//+------------------------------------------------------------------+
```

```
void ChangeArrowedLineEmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2)
  {
//--- ilk noktanın zamanı ayarlanmamışsa mevcut çubuk üzerinde olacak
   if(!time1)
      time1=TimeCurrent();
//--- ilk noktanın fiyatı ayarlanmamışsa mevcut Bid değerini alacak
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- ikinci noktanın zamanı ayarlanmamışsa, ilkinin 9 çubuk sonrasına konumlandırılır
   if(!time2)
     {
      //--- son 10 çubuğun açılış zamanını alacak bir dizi
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- ikinci noktayı birincisinden 9 çubuk ileri ayarla
      time2=temp[0];
     }
//--- ikinci noktanın fiyatı ayarlanmamışsa birinciye eşit olur
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
```

```
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- çizgiyi çizmek için noktaları tanımla
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- bir oklu çizgi oluştur
   if(!ArrowedLineCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, çizginin tutturma noktasını taşı
//--- döngü sayacı
   int v_steps=accuracy/5;
//--- ikinci tutturma noktasını dikey olarak taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p2<accuracy-1)
         p2+=1;
      //--- tutturma noktasını taşı
      if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- ilk tutturma noktasını dikey olarak taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p1>1)
         p1-=1;
```

```
      //--- tutturma noktasını taşı
      if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- yarım saniyelik gecikme
   Sleep(500);
//--- döngü sayacı
   int h_steps=bars/2;
//--- iki tutturma noktasını aynı anda taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- şu değerleri kullan
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- noktaları kaydır
      if(!ArrowedLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ArrowedLinePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.03 saniyelik gecikme
      Sleep(30);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- bir oklu çizgiyi sil
   ArrowedLineDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_CHANNEL

Eşit-Aralıklı Kanal

Not

Bir eşit-aralıklı kanalın sağa ve/veya sola doğru olan sürekliliklerinin (sırasıyla OBJPROP_RAY_RIGHT ve OBJPROP_RAY_LEFT özelliklerinin belirtilmesi mümkündür. Ayrıca, kanalın renk ile doldurulması modu da ayarlanabilir durumdadır.

Örnek

Aşağıdaki script, bir eşit-aralıklı kanal oluşturur ve onu çizelge üzerinde taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script \"Eşit-Aralıklı Kanal\" grafiksel nesnesini çizer."
#property description "Tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="Channel";   // Kanal ismi
input int             InpDate1=25;         // 1-inci noktanın tarihi, %
input int             InpPrice1=60;        // 1-inci noktanın fiyatı, %
input int             InpDate2=65;         // 2-inci noktanın tarihi, %
input int             InpPrice2=80;        // 2-inci noktanın fiyatı, %
input int             InpDate3=30;         // 3-üncü noktanın tarihi, %
```

```
input int              InpPrice3=40;         // 3-üncü noktanın fiyatı, %
input color            InpColor=clrRed;      // Kanal rengi
input ENUM_LINE_STYLE  InpStyle=STYLE_DASH;  // Kanal çizgilerinin stili
input int              InpWidth=2;           // Kanal çizgisi genişliği
input bool             InpBack=false;        // Arkaplan nesnesi
input bool             InpFill=false;        // Kanalı renkle doldurma
input bool             InpSelection=true;    // Taşımak için vurgula
input bool             InpRayLeft=false;     // Kanalın sola sürekliliği
input bool             InpRayRight=false;    // Kanalın sağa sürekliliği
input bool             InpHidden=true;       // Nesne listesinde gizle
input long             InpZOrder=0;          // Fare tıklaması önceliği
//+------------------------------------------------------------------+
//| Verilen koordinatlarla bir eşit-aralıklı kanal oluştur           |
//+------------------------------------------------------------------+
bool ChannelCreate(const long            chart_ID=0,        // çizelge kimliği
                   const string          name="Channel",    // kanal ismi
                   const int             sub_window=0,      // alt-pencere indisi
                   datetime              time1=0,           // ilk noktanın zamanı
                   double                price1=0,          // ilk noktanın fiyatı
                   datetime              time2=0,           // ikinci nokta zamanı
                   double                price2=0,          // ikinci nokta fiyatı
                   datetime              time3=0,           // üçüncü nokta zamanı
                   double                price3=0,          // üçüncü nokta fiyatı
                   const color           clr=clrRed,        // kanal rengi
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // kanal çizgilerinin sti
                   const int             width=1,           // kanal çizgilerinin gen
                   const bool            fill=false,        // kanalı renkle doldurma
                   const bool            back=false,        // arka-planda
                   const bool            selection=true,    // taşıma için vurgula
                   const bool            ray_left=false,    // kanalın sola doğru sür
                   const bool            ray_right=false,   // kanalın sağa doğru sür
                   const bool            hidden=true,       // nesne listesinde gizle
                   const long            z_order=0)         // fare tıklaması için önc
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- hata değerini sıfırla
   ResetLastError();
//--- verilen koordinatlarda bir kanal oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_CHANNEL,sub_window,time1,price1,time2,price2,tir
     {
      Print(__FUNCTION__,
            ": eşit-aralıklı kanalın oluşturulması başarısız! Hata kodu = ",GetLastEr
      return(false);
     }
//--- kanal rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- kanal çizgilerinin stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
```

```
//--- kanal çizgilerinin genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- kanal doldurma modunu etkinleştir (true) veya devre dışı bırak (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- kanalın sola doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı k
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- kanalın sağa doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı k
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Kanalın tutturma noktasını taşı                                  |
//+------------------------------------------------------------------+
bool ChannelPointChange(const long   chart_ID=0,     // çizelge kimliği
                        const string name="Channel", // kanal ismi
                        const int    point_index=0,  // tutturma noktası indisi
                        datetime     time=0,         // tutturma noktası zaman koordin
                        double       price=0)        // tutturma noktası fiyat koordin
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

```
//+------------------------------------------------------------------+
//| Kanalı sil                                                       |
//+------------------------------------------------------------------+
bool ChannelDelete(const long   chart_ID=0,     // çizelge kimliği
                   const string name="Channel") // kanal ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- kanalı sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": kanalın silinmesi başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------------+
//| Kanalın tutturma noktası değerlerini kontrol et ve varsayılan değerleri |
//| boş değerler için ayarla                                               |
//+------------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- İkinci noktanın (sağ) zamanı ayarlanmamışsa, ilk çubuk üzerinde olacak
   if(!time2)
      time2=TimeCurrent();
//--- İkinci noktanın fiyatı ayarlanmamışsa, Bid (alış) değerini alacak
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- İlk noktanın (sol) zamanı ayarlanmamışsa, ikinciden 9 çubuk sonra olacak
   if(!time1)
     {
      //--- son 10 çubuğun açılış zamanını alacak bir dizi
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- ilk noktayı ikinciden 9 çubuk sonraya ayarla
      time1=temp[0];
     }
//--- ilk noktanın fiyatı ayarlanmamışsa, ikincinin 300 puan üstüne taşı
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- üçüncü noktanın zamanı ayarlanmamışsa, birinci noktanın üstüne gelsin
   if(!time3)
      time3=time1;
//--- üçüncü noktanın fiyatı ayarlanmamışsa, ikinci noktanın fiyatına eşittir
   if(!price3)
      price3=price2;
```

```
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   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("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- kanalın çizileceği noktaları tanımla
   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;
//--- eşit-aralıklı kanalı oluştur
   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;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, kanalın tutturma noktalarını taşı
//--- döngü sayacı
   int h_steps=bars/6;
//--- ikinci tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d2<bars-1)
         d2+=1;
      //--- tutturma noktasını taşı
      if(!ChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- ilk tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d1>1)
         d1-=1;
      //--- tutturma noktasını taşı
      if(!ChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
```

```
   int v_steps=accuracy/10;
//--- üçüncü tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p3>1)
         p3-=1;
      //--- tutturma noktasını taşı
      if(!ChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- kanalı çizelgeden sil
   ChannelDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_STDDEVCHANNEL

Standart Sapma Kanalı.

Not

Standart Sapma Kanalının, sağa ve/veya sola doğru olan sürekliliklerinin (sırasıyla OBJPROP_RAY_RIGHT ve OBJPROP_RAY_LEFT özelliklerinin) belirtilmesi mümkündür. Ayrıca, kanalın renk ile doldurulması modu da ayarlanabilir durumdadır.

OBJPROP_DEVIATION özelliği, kanalın sapma değerini değiştirmek için kullanılır.

Örnek

Aşağıdaki script, Standart Sapma Kanalını çizelge üzerinde oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Standart Sapma Kanalı\" grafiksel nesnesini oluşturur
#property description "Tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="StdDevChannel";   // Kanal ismi
input int             InpDate1=10;               // 1-inci noktanın tarihi, %
input int             InpDate2=40;               // 2-inci noktanın tarihi, %
input double          InpDeviation=1.0;          // Sapma
input color           InpColor=clrRed;           // Kanal rengi
```

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Kanal çizgilerinin stili
input int             InpWidth=2;                // Kanal çizgilerinin genişliği
input bool            InpFill=false;             // Kanalın renk ile doldurulması
input bool            InpBack=false;             // Arkaplan kanalı
input bool            InpSelection=true;         // Taşımak için vurgula
input bool            InpRayLeft=false;          // Kanalın sola sürekliliği
input bool            InpRayRight=false;         // Kanalın sağa sürekliliği
input bool            InpHidden=true;            // Nesne listesinde gizle
input long            InpZOrder=0;               // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Verilen koordinatlarda standart sapma kanalını oluştur           |
//+------------------------------------------------------------------+
bool StdDevChannelCreate(const long             chart_ID=0,        // çizelge kimliğ
                         const string           name="Channel",    // kanal ismi
                         const int              sub_window=0,      // alt-pencere indis
                         datetime               time1=0,           // ilk noktanın zama
                         datetime               time2=0,           // ikinci noktanın z
                         const double           deviation=1.0,     // sapma
                         const color            clr=clrRed,        // kanal rengi
                         const ENUM_LINE_STYLE  style=STYLE_SOLID, // kanal çizgilerini
                         const int              width=1,           // kanal çizgilerini
                         const bool             fill=false,        // kanalın renk ile
                         const bool             back=false,        // arkaplanda
                         const bool             selection=true,    // taşıma için vurgu
                         const bool             ray_left=false,    // kanalın sola doğr
                         const bool             ray_right=false,   // kanalın sağa doğr
                         const bool             hidden=true,       // nesne listesinde
                         const long             z_order=0)         // fare tıklaması iç
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeChannelEmptyPoints(time1,time2);
//--- hata değerini sıfırla
   ResetLastError();
//--- verilen koordinatlarda bir kanal oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_STDDEVCHANNEL,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            ": standart sapma kanalı oluşturulamadı! Hata kodu = ",GetLastError());
      return(false);
     }
//--- standart sapma değeri kanalın genişliğini etkiler
   ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation);
//--- kanal rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- kanal çizgilerinin stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- kanal çizgilerinin genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- kanal doldurma modunu etkinleştir (true) veya devre dışı bırak (false)
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- kanalın sola doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı k
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- kanalın sağa doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı k
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Kanalın tutturma noktasını taşı                                  |
//+------------------------------------------------------------------+
bool StdDevChannelPointChange(const long   chart_ID=0,      // çizelge kimliği
                              const string name="Channel", // kanal ismi
                              const int    point_index=0,  // tutturma noktası indisi
                              datetime     time=0)         // tutturma noktası zaman k
  {
//--- eğer noktanın zamanı ayarlanmamışsa, noktayı şimdiki çubuğa taşı
   if(!time)
      time=TimeCurrent();
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Kanalın sapma değerini değiştir                                  |
//+------------------------------------------------------------------+
bool StdDevChannelDeviationChange(const long   chart_ID=0,      // çizelge kimliği
                                  const string name="Channel", // kanalın ismi
                                  const double deviation=1.0)  // sapma
```

```
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- trend çizgisinin eğim açısını değiştir
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_DEVIATION,deviation))
     {
      Print(__FUNCTION__,
            ": kanalın sapma değeri değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Kanalı sil                                                       |
//+------------------------------------------------------------------+
bool StdDevChannelDelete(const long   chart_ID=0,     // çizelge kimliği
                         const string name="Channel") // kanal ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- kanalı sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": kanalın silinmesi başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------------+
//| Kanalın tutturma noktası değerlerini kontrol et ve varsayılan değerleri |
//| boş değerler için ayarla                                               |
//+------------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,datetime &time2)
  {
//--- ikinci noktanın zamanı ayarlanmamışsa, mevcut çubuğun üstünde olacak
   if(!time2)
      time2=TimeCurrent();
//--- ilk noktanın zamanı ayarlanmamışsa, ikinci noktanın 9 çubuk solunda olacak
   if(!time1)
     {
      //--- son 10 çubuğun açılış zamanını alacak bir dizi
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- ilk noktayı ikinciden 9 çubuk sonraya ayarla
      time1=temp[0];
     }
```

```
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- kanalın çizileceği noktaları tanımla
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- standart sapma kanalını oluştur
   if(!StdDevChannelCreate(0,InpName,0,date[d1],date[d2],InpDeviation,InpColor,InpStyl
      InpWidth,InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
```

```
   ChartRedraw();
   Sleep(1000);
//--- şimdi, yatay olarak sağa doğru genişlet
//--- döngü sayacı
   int h_steps=bars/2;
//--- kanalı taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- şu değerleri kullan
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- tutturma noktalarını taşı
      if(!StdDevChannelPointChange(0,InpName,0,date[d1]))
         return;
      if(!StdDevChannelPointChange(0,InpName,1,date[d2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   double v_steps=InpDeviation*2;
//--- kanalı genişlet
   for(double i=InpDeviation;i<v_steps;i+=10.0/accuracy)
     {
      if(!StdDevChannelDeviationChange(0,InpName,i))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- kanalı çizelgeden sil
   StdDevChannelDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
```

```
}
```

OBJ_REGRESSION

Doğrusal Regresyon Kanalı.

Not

Doğrusal Regresyon Kanalının, sağa ve/veya sola doğru olan sürekliliklerinin (sırasıyla OBJPROP_RAY_RIGHT ve OBJPROP_RAY_LEFT özelliklerinin) belirtilmesi mümkündür. Ayrıca, kanalın renk ile doldurulması modu da ayarlanabilir durumdadır.

Örnek

Aşağıdaki script, Doğrusal Regresyon Kanalını çizelge üzerinde oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Doğrusal Regresyon Kanalı\" grafiksel nesnesini oluşt
#property description "Tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="Regression"; // Kanal ismi
input int             InpDate1=10;          // 1-inci noktanın tarihi, %
input int             InpDate2=40;          // 2-inci noktanın tarihi, %
input color           InpColor=clrRed;      // Kanal rengi
input ENUM_LINE_STYLE InpStyle=STYLE_DASH;  // Kanal çizgisi Stili
input int             InpWidth=2;           // Kanal çizgisi genişliği
```

```
input bool             InpFill=false;        // Kanalın renkle doldurulması
input bool             InpBack=false;        // Arka-plan rengi
input bool             InpSelection=true;    // Taşıma için vurgula
input bool             InpRayLeft=false;     // Kanalın sola doğru sürekliliği
input bool             InpRayRight=false;    // Kanalın sağa doğru sürekliliği
input bool             InpHidden=true;       // Nesne listesinde gizle
input long             InpZOrder=0;          // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Doğrusal Regresyon Kanalını verilen koordinatlarda oluştur       |
//+------------------------------------------------------------------+
bool RegressionCreate(const long             chart_ID=0,        // çizelge kimliği
                      const string           name="Regression", // kanal ismi
                      const int              sub_window=0,      // alt-pencere indisi
                      datetime               time1=0,           // ilk nokta zamanı
                      datetime               time2=0,           // ikinci nokta zamanı
                      const color            clr=clrRed,        // kanal rengi
                      const ENUM_LINE_STYLE  style=STYLE_SOLID, // kanal çizgilerinin s
                      const int              width=1,           // kanal çizgilerinin ç
                      const bool             fill=false,        // kanalı renk ile dolc
                      const bool             back=false,        // arka-planda
                      const bool             selection=true,    // taşıma için vurgula
                      const bool             ray_left=false,    // kanalın sola doğru s
                      const bool             ray_right=false,   // kanalın sağa doğru s
                      const bool             hidden=true,       // nesne listesinde giz
                      const long             z_order=0)         // fare tıklaması öncel
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeRegressionEmptyPoints(time1,time2);
//--- hata değerini sıfırla
   ResetLastError();
//--- verilen koordinatlarda bir kanal oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_REGRESSION,sub_window,time1,0,time2,0))
     {
      Print(__FUNCTION__,
            ": doğrusal regresyon kanalı oluşturulamadı! Hata kodu = ",GetLastError());
      return(false);
     }
//--- kanal rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- kanal çizgilerinin stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- kanal çizgilerinin genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- kanal doldurma modunu etkinleştir (true) veya devre dışı bırak (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
```

```
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- kanalın sola doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı k
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- kanalın sağa doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı k
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Kanalın tutturma noktasını taşı                                  |
//+------------------------------------------------------------------+
bool RegressionPointChange(const long   chart_ID=0,     // çizelge kimliği
                           const string name="Channel", // kanal ismi
                           const int    point_index=0,  // tutturma noktası indisi
                           datetime     time=0)         // tutturma noktası zaman koo
  {
//--- eğer noktanın zamanı ayarlanmamışsa, noktayı şimdiki çubuğa taşı
   if(!time)
      time=TimeCurrent();
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,0))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Kanalı sil                                                       |
//+------------------------------------------------------------------+
bool RegressionDelete(const long   chart_ID=0,     // çizelge kimliği
                      const string name="Channel") // kanal ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- kanalı sil
   if(!ObjectDelete(chart_ID,name))
     {
```

```
      Print(__FUNCTION__,
            ": kanalın silinmesi başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------------+
//| Kanalın tutturma noktası değerlerini kontrol et ve varsayılan değerleri |
//| boş değerler için ayarla                                               |
//+------------------------------------------------------------------------+
void ChangeRegressionEmptyPoints(datetime &time1,datetime &time2)
  {
//--- ikinci noktanın zamanı ayarlanmamışsa, mevcut çubuğun üstünde olacak
   if(!time2)
      time2=TimeCurrent();
//--- ilk noktanın zamanı ayarlanmamışsa, ikinci noktanın 9 çubuk solunda olacak
   if(!time1)
     {
      //--- son 10 çubuğun açılış zamanını alacak bir dizi
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- ilk noktayı ikinciden 9 çubuk sonraya ayarla
      time1=temp[0];
     }
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate1<0 || InpDate1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
```

```
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- kanalın çizileceği noktaları tanımla
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
//--- doğrusal regresyon kanalını oluştur
   if(!RegressionCreate(0,InpName,0,date[d1],date[d2],InpColor,InpStyle,InpWidth,
      InpFill,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, kanalı yatay olarak sağa doğru taşı
//--- döngü sayacı
   int h_steps=bars/2;
//--- kanalı taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- şu değerleri kullan
      if(d1<bars-1)
         d1+=1;
      if(d2<bars-1)
         d2+=1;
      //--- tutturma noktalarını taşı
      if(!RegressionPointChange(0,InpName,0,date[d1]))
         return;
      if(!RegressionPointChange(0,InpName,1,date[d2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
```

```
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- kanalı çizelgeden sil
   RegressionDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_PITCHFORK

Andrews Dirgeni (Üçlü Çizgi).

Not

"Andrews Dirgeni" (Üçlü Çizgi) için, sağa ve/veya sola doğru olan sürekliliklerinin (sırasıyla OBJPROP_RAY_RIGHT ve OBJPROP_RAY_LEFT özelliklerinin) belirtilmesi mümkündür.

Ayrıca, seviye çizgilerinin sayısını, değerlerini ve renklerini de belirtebilirsiniz.

Örnek

Aşağıdaki script, Andrews Dirgenini çizelge üzerinde oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Andrews Dirgeni\" grafiksel nesnesini çizer."
#property description "Tutturma (çapa) noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="Pitchfork";  // Dirgenin (üçlü çizginin) ismi
input int             InpDate1=14;          // 1-inci noktanın tarihi, %
input int             InpPrice1=40;         // 1-inci noktanın fiyatı, %
input int             InpDate2=18;          // 2-inci noktanın tarihi, %
input int             InpPrice2=50;         // 2-inci noktanın fiyatı, %
input int             InpDate3=18;          // 3-üncü noktanın tarihi, %
input int             InpPrice3=30;         // 3-üncü noktanın fiyatı, %
```

```
input color           InpColor=clrRed;      // Dirgen rengi
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID; // Dirgenin stili
input int             InpWidth=1;           // Dirgen çizgilerinin kalınlığı
input bool            InpBack=false;        // Arkaplan nesnesi
input bool            InpSelection=true;    // Taşıma için vurgula
input bool            InpRayLeft=false;     // Dirgenin sola doğru sürekliliği
input bool            InpRayRight=false;    // Dirgenin sağa doğru sürekliliği
input bool            InpHidden=true;       // Nesne listesinde gizle
input long            InpZOrder=0;          // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Andrews Dirgenini verilen koordinatlarda oluştur                 |
//+------------------------------------------------------------------+
bool PitchforkCreate(const long            chart_ID=0,        // çizelge kimliği
                     const string          name="Pitchfork",  // dirgen ismi
                     const int             sub_window=0,      // alt pencere indisi
                     datetime              time1=0,           // ilk nokta zamanı
                     double                price1=0,          // ilk nokta fiyatı
                     datetime              time2=0,           // ikinci nokta zamanı
                     double                price2=0,          // ikinci nokta fiyatı
                     datetime              time3=0,           // üçüncü nokta zamanı
                     double                price3=0,          // üçüncü nokta fiyatı
                     const color           clr=clrRed,        // dirgen çizgilerinin r
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // dirgen çizgilerinin s
                     const int             width=1,           // dirgen çizgilerinin k
                     const bool            back=false,        // arka-plan çizimi
                     const bool            selection=true,    // taşıma için vurgula
                     const bool            ray_left=false,    // dirgenin sola doğru s
                     const bool            ray_right=false,   // dirgenin sağa doğru s
                     const bool            hidden=true,       // nesne listesinde gizl
                     const long            z_order=0)         // fare tıklaması için ö
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- hata değerini sıfırla
   ResetLastError();
//--- Andrews Dirgenini verilen koordinatlarda oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_PITCHFORK,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            ": \"Andrews Dirgeni\" oluşturulamadı! Hata kodu = ",GetLastError());
      return(false);
     }
//--- rengi ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- çizgilerin genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- dirgenin sola doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- dirgenin sağa doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Andrews Çatalının seviyelerini ve parametrelerini ayarla         |
//+------------------------------------------------------------------+
bool PitchforkLevelsSet(int             levels,           // seviye çizgilerinin sayıs
                        double          &values[],        // seviye çizgilerinin değer
                        color           &colors[],        // seviye çizgilerinin rengi
                        ENUM_LINE_STYLE &styles[],        // seviye çizgilerinin stili
                        int             &widths[],        // seviye çizgilerinin kalın
                        const long      chart_ID=0,       // çizelge tanımlayıcı
                        const string    name="Pitchfork") // dirgen ismi
  {
//--- dizi büyüklüklerini ayarla
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": dizi uzunluğu seviyelerin sayısıyla örtüşmüyor, hata!");
      return(false);
     }
//--- seviyelerin sayısını ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- seviyelerin özeliklerini döngü içinde ayarla
   for(int i=0;i<levels;i++)
     {
      //--- seviye değeri
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- seviye rengi
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- seviye stili
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- seviye genişliği
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
```

```
      //--- seviye açıklaması
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Adrews Dirgeninin tutturma noktasını taşı                        |
//+------------------------------------------------------------------+
bool PitchforkPointChange(const long   chart_ID=0,       // çizelge kimliği
                          const string name="Pitchfork", // kanal ismi
                          const int    point_index=0,    // tutturma noktası indisi
                          datetime     time=0,           // tutturma noktası zaman koo
                          double       price=0)          // tutturma noktası fiyat koo
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Andrews Dirgenini sil                                            |
//+------------------------------------------------------------------+
bool PitchforkDelete(const long   chart_ID=0,       // çizelge kimliği
                     const string name="Pitchfork") // kanal ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- kanalı sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Andrews Dirgeni\" silinemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
```

```
  }
//+------------------------------------------------------------------+
//| Andrews Dirgeninin tutturma noktası değerlerini kontrol et ve    |
//| boş olanlar için varsayılan değerleri ayarla                     |
//+------------------------------------------------------------------+
void ChangeChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                              double &price2,datetime &time3,double &price3)
  {
//--- ikinci (sağ üst) noktanın zamanı ayarlanmamışsa, mevcut çubuk üstünde olacak
   if(!time2)
      time2=TimeCurrent();
//--- İkinci noktanın fiyatı ayarlanmamışsa, Bid (alış) değerini alacak
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- İlk noktanın (sol) zamanı ayarlanmamışsa, ikinciden 9 çubuk sonra olacak
   if(!time1)
     {
      //--- son 10 çubuğun açılış zamanını alacak bir dizi
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- ilk noktayı ikinciden 9 çubuk sonraya ayarla
      time1=temp[0];
     }
//--- ilk noktanın fiyatı ayarlanmamışsa, ikincinin 200 puan altına yerleştir
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- üçüncü noktanın zamanı ayarlanmamışsa, ikinci noktaya karşılık gelecek
   if(!time3)
      time3=time2;
//--- üçüncü noktanın fiyatı ayarlanmamışsa, ilkinin 200 puan aşağısına konumla
   if(!price3)
      price3=price1-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   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("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
```

```
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- Andrews Dirgenini çizmek için noktaları tanımla
   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;
//--- dirgeni oluştur
   if(!PitchforkCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pric
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, dirgenin tutturma noktasını taşı
//--- döngü sayacı
   int v_steps=accuracy/10;
//--- ilk tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p1>1)
         p1-=1;
      //--- tutturma noktasını taşı
```

```
      if(!PitchforkPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   int h_steps=bars/8;
//--- üçüncü tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d3<bars-1)
         d3+=1;
      //--- tutturma noktasını taşı
      if(!PitchforkPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   v_steps=accuracy/10;
//--- ikinci tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p2>1)
         p2-=1;
      //--- tutturma noktasını taşı
      if(!PitchforkPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
```

```
      }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- dirgeni çizelgeden sil
   PitchforkDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_GANNLINE

Gann Çizgisi.

Not

"Gann çizgilerinin" sağa ve/veya sola doğru olan sürekliliklerinin (sırasıyla OBJPROP_RAY_RIGHT ve OBJPROP_RAY_LEFT özelliklerinin) belirtilmesi mümkündür.

Ölçekli Gann açısının ve ikinci tutturma noktası koordinatlarının her ikisi de çizgi eğimini ayarlamak için kullanılabilir.

Örnek

Aşağıdaki script Gann Çizgisini çizelge üzerinde oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Gann çizgisi\" grafiksel nesnesini çizer."
#property description "Tutturma (çapa) noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="GannLine";        // Çizgi ismi
input int             InpDate1=20;               // 1-inci noktanın tarihi, %
input int             InpPrice1=75;              // 1-inci noktanın fiyatı, %
input int             InpDate2=80;               // 2-inci noktanın tarihi, %
input double          InpAngle=0.0;              // Gan Açısı
```

```
input double           InpScale=1.0;              // Ölçek
input color            InpColor=clrRed;           // Çizgi rengi
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // Çizgi stili
input int              InpWidth=2;                // Çizgi stili
input bool             InpBack=false;             // Arka plan çizgisi
input bool             InpSelection=true;         // Taşımak için vurgula
input bool             InpRayLeft=false;          // Çizginin sola doğru sürekliliği
input bool             InpRayRight=true;          // Çizginin sağa doğru sürekliliği
input bool             InpHidden=true;            // Nesne listesinde gizle
input long             InpZOrder=0;               // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Gann Çizgisini; koordinatlar, açı ve ölçek ile oluştur           |
//+------------------------------------------------------------------+
bool GannLineCreate(const long             chart_ID=0,        // çizelge kimliği
                    const string           name="GannLine",   // çizgi ismi
                    const int              sub_window=0,      // alt pencere indisi
                    datetime               time1=0,           // ilk nokta zamanı
                    double                 price1=0,          // ilk nokta fiyatı
                    datetime               time2=0,           // ikinci nokta zamanı
                    const double           angle=1.0,         // Gann açısı
                    const double           scale=1.0,         // ölçek
                    const color            clr=clrRed,        // çizgi rengi
                    const ENUM_LINE_STYLE  style=STYLE_SOLID, // çizgi stili
                    const int              width=1,           // çizgi kalınlığı
                    const bool             back=false,        // arka planda
                    const bool             selection=true,    // taşıma için vurgula
                    const bool             ray_left=false,    // çizginin sola doğru sü
                    const bool             ray_right=true,    // çizginin sağa doğru sü
                    const bool             hidden=true,       // nesneyi listede gizle
                    const long             z_order=0)         // fare tıklaması için ön
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeGannLineEmptyPoints(time1,price1,time2);
//--- hata değerini sıfırla
   ResetLastError();
//--- verilen koordinatlarla bir Gann Çizgisi oluştur
//--- ikinci tutturma noktasının doğru koordinatları,
//--- Gann açısının ve/veya ölçeğin değişiminden sonra otomatik olarak yeniden tanımla
   if(!ObjectCreate(chart_ID,name,OBJ_GANNLINE,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": \"Gann Çizgisi oluşturulamadı\"! Hata kodu = ",GetLastError());
      return(false);
     }
//--- Gann açısını değiştir
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- ölçeği değiştir (çubuk başına düşen pip olarak)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- çizgi rengini ayarla
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- çizgi görünüm stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- çizgi genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- çizginin sola doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- çizginin sağa doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Gann çizgisi çapa noktasını taşı                                 |
//+------------------------------------------------------------------+
bool GannLinePointChange(const long   chart_ID=0,      // çizelge kimliği
                         const string name="GannLine", // çizgi ismi
                         const int    point_index=0,   // tutturma noktası indisi
                         datetime     time=0,          // çapa noktası zaman koordinat
                         double       price=0)         // çapa noktası fiyat koordinat
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- çizginin tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
```

```
  }
//+------------------------------------------------------------------+
//| Gann açısını değiştir                                            |
//+------------------------------------------------------------------+
bool GannLineAngleChange(const long   chart_ID=0,      // çizelge kimliği
                         const string name="GannLine", // çizgi ismi
                         const double angle=1.0)       // Gann açısı
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- Gann açısını değiştir
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle))
     {
      Print(__FUNCTION__,
            ": Gann açısı değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Gann Çizgisinin ölçeğini değiştir                                |
//+------------------------------------------------------------------+
bool GannLineScaleChange(const long   chart_ID=0,      // çizelge kimliği
                         const string name="GannLine", // çizgi ismi
                         const double scale=1.0)       // ölçek
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- ölçeği değiştir (çubuk başına düşen pip olarak)
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
     {
      Print(__FUNCTION__,
            ": ölçek değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fonksiyon, Gann çizgisini çizelgeden siler                       |
//+------------------------------------------------------------------+
bool GannLineDelete(const long   chart_ID=0,      // çizelge kimliği
                    const string name="GannLine") // çizgi ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- Gann çizgisini sil
   if(!ObjectDelete(chart_ID,name))
```

```
      {
       Print(__FUNCTION__,
             ": \"Gann Çizgisi\" silinemedi! Hata kodu = ",GetLastError());
       return(false);
      }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Gann Çizgisinin tutturma noktası değerlerini kontrol et ve       |
//| boş olanlar için varsayılan değerleri kullan                     |
//+------------------------------------------------------------------+
void ChangeGannLineEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- ikinci noktanın zamanı ayarlanmamışsa, mevcut çubuğun üstünde olacak
   if(!time2)
      time2=TimeCurrent();
//--- ilk noktanın zamanı ayarlanmamışsa, ikinci noktanın 9 çubuk solunda olacak
   if(!time1)
     {
      //--- son 10 çubuğun açılış zamanını alacak bir dizi
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- ilk noktayı ikinciden 9 çubuk sonraya ayarla
      time1=temp[0];
     }
//--- ilk noktanın fiyatı ayarlanmamışsa mevcut Bid değerini alacak
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
```

```
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- Gan çizgisini çizmek için noktaları tanımla
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- Gann Çizgisini oluştur
   if(!GannLineCreate(0,InpName,0,date[d1],price[p1],date[d2],InpAngle,InpScale,InpCol
      InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, tutturma noktasını taşı ve açıyı değiştir
//--- döngü sayacı
   int v_steps=accuracy/2;
//--- ilk tutturma noktasını dikey olarak taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p1>1)
         p1-=1;
      //--- tutturma noktasını taşı
      if(!GannLinePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
```

```
      }
//--- yarım saniyelik gecikme
   Sleep(500);
//--- Gan açısının (ilk tutturma noktasının taşınmasıyla değişen)
//--- mevcut değerini tanımla
   double curr_angle;
   if(!ObjectGetDouble(0,InpName,OBJPROP_ANGLE,0,curr_angle))
      return;
//--- döngü sayacı
   v_steps=accuracy/8;
//--- Gann açısını değiştir
   for(int i=0;i<v_steps;i++)
     {
      if(!GannLineAngleChange(0,InpName,curr_angle-0.05*i))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- çizgiyi çizelgeden sil
   GannLineDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_GANNFAN

Gann Yelpazesi.

Not

Gann Yelpazesi için, ENUM_GANN_DIRECTION sayımıyla trend tipi belirlemek mümkündür. Ölçek değerini (OBJPROP_SCALE) ayarlamak için, fan çizgilerinin eğim açıları değiştirilebilir.

Örnek

Aşağıdaki script, Gann Yelpazesini çizelge üzerinde oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Gann Yelpazesi\" grafiksel nesnesini oluşturur."
#property description "Tutturma (çapa) noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="GannFan";         // Yelpaze ismi
input int             InpDate1=15;               // 1-inci noktanın tarihi, %
input int             InpPrice1=25;              // 1-inci noktanın fiyatı, %
input int             InpDate2=85;               // 2-inci noktanın tarihi, %
input double          InpScale=2.0;              // Ölçek
input bool            InpDirection=false;        // Trend yönü
input color           InpColor=clrRed;           // Yelpaze rengi
```

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Yelpaze çizgilerinin rengi
input int             InpWidth=1;                // Yelpaze çizgilerinin kalınlığı
input bool            InpBack=false;             // Arka plan nesnesi
input bool            InpSelection=true;         // Taşımak için vurgula
input bool            InpHidden=true;            // Nesne listesinde gizle
input long            InpZOrder=0;               // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Gann Yelpazesini oluştur                                         |
//+------------------------------------------------------------------+
bool GannFanCreate(const long            chart_ID=0,        // çizelge kimliği
                   const string          name="GannFan",    // yelpaze ismi
                   const int             sub_window=0,      // alt pencere indisi
                   datetime              time1=0,           // ilk noktanın zamanı
                   double                price1=0,          // ilk noktanın fiyatı
                   datetime              time2=0,           // ikinci nokta zamanı
                   const double          scale=1.0,         // ölçek
                   const bool            direction=true,    // trend yönü
                   const color           clr=clrRed,        // yelpaze rengi
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // yelpaze çizgilerinin st
                   const int             width=1,           // fan çizgilerinin kalını
                   const bool            back=false,        // arka-planda
                   const bool            selection=true,    // taşıma için vurgula
                   const bool            hidden=true,       // nesne listesinde gizle
                   const long            z_order=0)         // fare tıklaması için önc
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeGannFanEmptyPoints(time1,price1,time2);
//--- hata değerini sıfırla
   ResetLastError();
//--- verilmiş koordinatlarda Gann Yelpazesini oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_GANNFAN,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": \"Gann Yelpazesinin \" oluşturulması başarısız! Hata kodu = ",GetLastEr
      return(false);
     }
//--- ölçeği değiştir (çubuk başına düşen pip olarak)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- Gann Yelpazesinin trend yönünü değiştir (true - azalan, false - artan)
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- Yelpaze rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- Yelpaze çizgilerinin görüntülenme stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- Yelpaze çizgilerinin kalınlığını ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
```

```
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Gann Yelpazesinin tutturma noktasını taşı                        |
//+------------------------------------------------------------------+
bool GannFanPointChange(const long   chart_ID=0,     // çizelge kimliği
                        const string name="GannFan", // yelpaze ismi
                        const int    point_index=0,  // tutturma noktası indisi
                        datetime     time=0,         // tutturma noktası zaman koordi
                        double       price=0)        // tutturma noktası fiyat koordi
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğ
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- yelpazenin tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Gann Yelpazesinin ölçeğini değiştir                              |
//+------------------------------------------------------------------+
bool GannFanScaleChange(const long   chart_ID=0,     // çizelge kimliği
                        const string name="GannFan", // yelpaze ismi
                        const double scale=1.0)      // ölçek
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- ölçeği değiştir (çubuk başına düşen pip olarak)
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
```

```
     {
      Print(__FUNCTION__,
            ": ölçek değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Gann Yelpazesinin trend yönünü değiştir                          |
//+------------------------------------------------------------------+
bool GannFanDirectionChange(const long   chart_ID=0,      // çizelge kimliği
                            const string name="GannFan", // yelpaze ismi
                            const bool   direction=true) // trend yönü
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- Gann Yelpazesinin trend yönünü değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            ": trend yönü değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fonksiyon Gann Izgarasını çizelgeden siler                       |
//+------------------------------------------------------------------+
bool GannFanDelete(const long   chart_ID=0,      // çizelge kimliği
                   const string name="GannFan") // yelpaze ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- Gann Yelpazesini sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Gann Yelpazesi\" silinemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
// | Gann Yelpazesinin tutturma noktası değerlerini kontrol et ve    |
//| boş olanlar için varsayılan değerleri kullan                     |
//+------------------------------------------------------------------+
```

```
void ChangeGannFanEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- ikinci noktanın zamanı ayarlanmamışsa, mevcut çubuğun üstünde olacak
   if(!time2)
      time2=TimeCurrent();
//--- ilk noktanın zamanı ayarlanmamışsa, ikinci noktanın 9 çubuk solunda olacak
   if(!time1)
     {
      //--- son 10 çubuğun açılış zamanını alacak bir dizi
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- ilk noktayı ikinciden 9 çubuk sonraya ayarla
      time1=temp[0];
     }
//--- ilk noktanın fiyatı ayarlanmamışsa mevcut Bid değerini alacak
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
```

```
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- Gann Yelpazesini çizmek için noktaları tanımla
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- Gann Yelpazesi
   if(!GannFanCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- Yelpazenin tutturma noktasını taşı
//--- döngü sayacı
   int v_steps=accuracy/2;
//--- ilk tutturma noktasını dikey olarak taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p1<accuracy-1)
         p1+=1;
      //--- tutturma noktasını taşı
      if(!GannFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- yelpazenin trend yönünü azalan olarak ayarla
   GannFanDirectionChange(0,InpName,true);
//--- çizelgeyi yeniden çiz
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- yelpazeyi çizelgeden sil
   GannFanDelete(0,InpName);
   ChartRedraw();
```

```
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_GANNGRID

Gann Izgarası.

Not

Gann Izgarası için, ENUM_GANN_DIRECTIONsayımıyla trend tipini belirlemek mümkündür. Ölçek değerini (OBJPROP_SCALE) ayarlayarak, eğim açısını ve ızgarayı değiştirmek mümkündür.

Örnek

Aşağıdaki script, çizelge üzerinde Gann Izgarası oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script \"Gann Izgarası\" grafiksel nesnesini oluşturur."
#property description "Tutturma (çapa) noktası koordinatları, çizelge penceresi""
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="GannGrid";        // Izgaranın ismi
input int             InpDate1=15;               // 1-inci noktanın tarihi, %
input int             InpPrice1=25;              // 1-inci noktanın fiyatı, %
input int             InpDate2=35;               // 2-inci noktanın tarihi, %
input double          InpScale=3.0;              // Ölçek
input bool            InpDirection=false;        // Trend yönü
input color           InpColor=clrRed;           // Izgara rengi
```

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Izgara çizgilerinin stili
input int             InpWidth=1;                // Yelpaze çizgilerinin kalınlığı
input bool            InpBack=false;             // Arkaplan nesnesi
input bool            InpSelection=true;         // Taşımak için vurgula
input bool            InpHidden=true;            // Nesne listesinde gizle
input long            InpZOrder=0;               // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Gan Izgarası Oluştur                                             |
//+------------------------------------------------------------------+
bool GannGridCreate(const long            chart_ID=0,        // çizelge kimliği
                    const string          name="GannGrid",   // ızgara ismi
                    const int             sub_window=0,      // alt pencere indisi
                    datetime              time1=0,           // ilk nokta zamanı
                    double                price1=0,          // ilk nokta fiyatı
                    datetime              time2=0,           // ikinci nokta zamanı
                    const double          scale=1.0,         // ölçek
                    const bool            direction=true,    // trend yönü
                    const color           clr=clrRed,        // ızgara rengi
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // ızgara çizgilerinin st
                    const int             width=1,           // ızgara çizgilerinin ka
                    const bool            back=false,        // arka planda
                    const bool            selection=true,    // taşıma için vurgula
                    const bool            hidden=true,       // nesneyi listede gizle
                    const long            z_order=0)         // fare tıklaması için ön
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeGannGridEmptyPoints(time1,price1,time2);
//--- hata değerini sıfırla
   ResetLastError();
//--- Gann Izgarasını verilen koordinatlara göre oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_GANNGRID,sub_window,time1,price1,time2,0))
     {
      Print(__FUNCTION__,
            ": \"Gann Izgarası\" oluşturulamadı! Hata kodu = ",GetLastError());
      return(false);
     }
//--- ölçeği değiştir (çubuk başına düşen pip olarak)
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- Gann Yelpazesinin trend yönünü değiştir (true - azalan, false - artan)
   ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction);
//--- ızgara rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- ızgara çizgilerinin stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- ızgara çizgilerinin kalınlığını ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
```

```
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Gann Izgarası tutturma noktasını taşı                            |
//+------------------------------------------------------------------+
bool GannGridPointChange(const long   chart_ID=0,      // çizelge kimliği
                         const string name="GannGrid", // ızgara ismi
                         const int    point_index=0,   // tutturma noktası indisi
                         datetime     time=0,          // çapa noktası zaman koordinat
                         double       price=0)         // çapa noktası fiyat koordinat
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğ
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- ızgara tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Gann Izgarasının ölçeğini değiştir                               |
//+------------------------------------------------------------------+
bool GannGridScaleChange(const long   chart_ID=0,      // çizelge kimliği
                         const string name="GannGrid", // ızgara ismi
                         const double scale=1.0)       // ölçek
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- ölçeği değiştir (çubuk başına düşen pip olarak)
   if(!ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale))
```

```
     {
      Print(__FUNCTION__,
            ": ölçek değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Gann Izgarasının trend yönünü değiştir                           |
//+------------------------------------------------------------------+
bool GannGridDirectionChange(const long   chart_ID=0,      // çizelge kimliği
                             const string name="GannGrid", // çizelge ismi
                             const bool   direction=true)  // trend yönü
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- Gann Izgarasının trend yönünü değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_DIRECTION,direction))
     {
      Print(__FUNCTION__,
            ": trend yönü değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fonksiyon Gann Izgarasını çizelgeden siler                       |
//+------------------------------------------------------------------+
bool GannGridDelete(const long   chart_ID=0,      // çizelge kimliği
                    const string name="GannGrid") // ızgara ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- Gann Izgarasını sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Gann Izgarasının\" silinmesi başarısız oldu! Hata kodu = ",GetLastEr
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Gann Izgarasının tutturma noktası değerlerini kontrol et ve      |
//| boş olanlar için varsayılan değerleri kullan                     |
//+------------------------------------------------------------------+
```

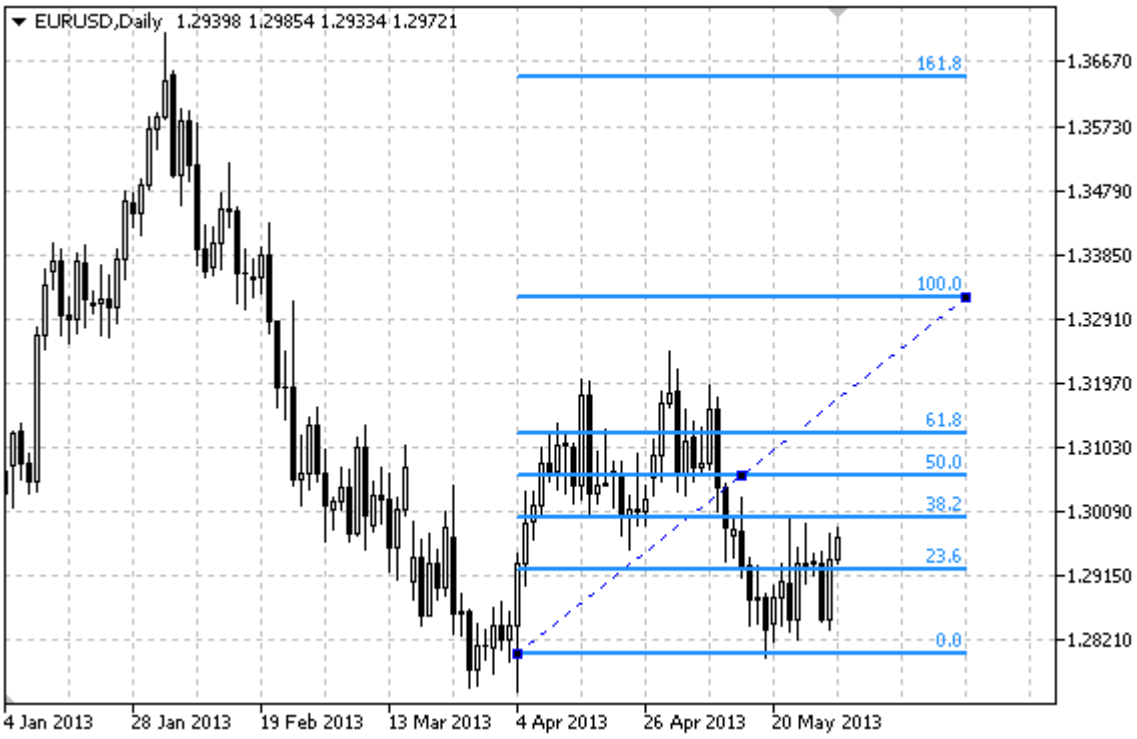
```
void ChangeGannGridEmptyPoints(datetime &time1,double &price1,datetime &time2)
  {
//--- ikinci noktanın zamanı ayarlanmamışsa, mevcut çubuğun üstünde olacak
   if(!time2)
      time2=TimeCurrent();
//--- ilk noktanın zamanı ayarlanmamışsa, ikinci noktanın 9 çubuk solunda olacak
   if(!time1)
     {
      //--- son 10 çubuğun açılış zamanını alacak bir dizi
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- ilk noktayı ikinciden 9 çubuk sonraya ayarla
      time1=temp[0];
     }
//--- ilk noktanın fiyatı ayarlanmamışsa mevcut Bid değerini alacak
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
```

```
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- Gann Izgarasının çizileceği noktaları tanımla
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
//--- Gann Izgarasını oluştur
   if(!GannGridCreate(0,InpName,0,date[d1],price[p1],date[d2],InpScale,InpDirection,
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, ızgaranın tutturma noktasını taşı
//--- döngü sayacı
   int v_steps=accuracy/4;
//--- ilk tutturma noktasını dikey olarak taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p1<accuracy-1)
         p1+=1;
      if(!GannGridPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   int h_steps=bars/4;
//--- ikinci tutturma noktasını yatay olarak taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d2<bars-1)
         d2+=1;
      if(!GannGridPointChange(0,InpName,1,date[d2],0))
         return;
```

```
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- ızgaranın trend yönünü azalan olarak ayarla
   GannGridDirectionChange(0,InpName,true);
//--- çizelgeyi yeniden çiz
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- ızgarayı çizelgeden siler
   GannGridDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_FIBO

Fibonacci Düzeltme Seviyeleri.



Not

"Fibonacci Düzeltme Seviyelerinin" sağa ve/veya sola doğru olan sürekliliklerinin (sırasıyla OBJPROP_RAY_RIGHT ve OBJPROP_RAY_LEFT özelliklerinin) belirtilmesi mümkündür.

Ayrıca, seviye çizgilerinin sayısını, değerlerini ve renklerini de belirtebilirsiniz.

Örnek

Aşağıdaki script, çizelge üzerinde Fibonacci Düzeltme Seviyelerini oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Fibonacci Düzeltme Seviyeleri\" grafiksel nesnesini
#property description "Tutturma (çapa) noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="FiboLevels";      // Nesne ismi
input int             InpDate1=10;               // 1-inci noktanın tarihi, %
input int             InpPrice1=65;              // 1-inci noktanın fiyatı, %
input int             InpDate2=90;               // 2-inci noktanın tarihi, %
input int             InpPrice2=85;              // 2-inci noktanın fiyatı, %
input color           InpColor=clrRed;           // Nesne rengi
```

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Çizgi stili
input int             InpWidth=2;                // Çizgi stili
input bool            InpBack=false;             // Arka-plan nesnesi
input bool            InpSelection=true;         // Taşımak için vurgula
input bool            InpRayLeft=false;          // Nesnenin sola sürekliliği
input bool            InpRayRight=false;         // Nesnenin sağa sürekliliği
input bool            InpHidden=true;            // Nesne listesinde gizle
input long            InpZOrder=0;               // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Fibonacci Düzeltme Seviyelerini verilen koordinatlarda oluştur   |
//+------------------------------------------------------------------+
bool FiboLevelsCreate(const long            chart_ID=0,        // çizelge tanımlayıcıs
                      const string          name="FiboLevels", // nesne ismi
                      const int             sub_window=0,      // alt-pencere indisi
                      datetime              time1=0,           // ilk nokta zamanı
                      double                price1=0,          // ilk nokta fiyatı
                      datetime              time2=0,           // ikinci nokta zamanı
                      double                price2=0,          // ikinci nokta fiyatı
                      const color           clr=clrRed,        // nesne rengi
                      const ENUM_LINE_STYLE style=STYLE_SOLID, // nesnenin çizgi stili
                      const int             width=1,           // nesnenin çizgi genis
                      const bool            back=false,        // arka-planda
                      const bool            selection=true,    // taşıma için vurgula
                      const bool            ray_left=false,    // nesnenin sola sürekl
                      const bool            ray_right=false,   // object's nesnenin sa
                      const bool            hidden=true,       // nesne listesinde giz
                      const long            z_order=0)         // fare tıklaması öncel
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeFiboLevelsEmptyPoints(time1,price1,time2,price2);
//--- hata değerini sıfırla
   ResetLastError();
//--- Verilen koordinatlarda Fibonacci Düzeltme Seviyelerini oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_FIBO,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": \"Fibonacci Düzeltme Seviyeleri\" oluşturulamadı! Hata kodu = ",GetLast
      return(false);
     }
//--- rengi ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- çizgi genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
```

```
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesnenin sola doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- nesnenin sağa doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Seviyelerin parametrelerini ve sayılarını ayarla                 |
//+------------------------------------------------------------------+
bool FiboLevelsSet(int              levels,           // seviye çizgilerinin sayısı
                   double           &values[],        // seviye çizgilerinin değerleri
                   color            &colors[],        // seviye çizgilerinin rengi
                   ENUM_LINE_STYLE  &styles[],        // seviye çizgilerinin stili
                   int              &widths[],        // seviye çizgilerinin kalınlığı
                   const long       chart_ID=0,       // çizelge tanımlayıcısı
                   const string     name="FiboLevels") // nesne ismi
  {
//--- dizi büyüklüklerini ayarla
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) || levels!=ArraySize(widt
     {
      Print(__FUNCTION__,": dizi uzunluğu seviyelerin sayısıyla örtüşmüyor, hata!");
      return(false);
     }
//--- seviyelerin sayısını ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- seviyelerin özeliklerini döngü içinde ayarla
   for(int i=0;i<levels;i++)
     {
      //--- seviye değeri
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- seviye rengi
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- seviye stili
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- seviye genişliği
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- seviye açıklaması
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- başarılı çalıştırma
```

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Fibonacci Düzeltme Seviyelerinin tutturma noktalarını taşı       |
//+------------------------------------------------------------------+
bool FiboLevelsPointChange(const long   chart_ID=0,        // çizelge tanımlayıcısı
                           const string name="FiboLevels", // nesne ismi
                           const int    point_index=0,     // tutturma noktası indisi
                           datetime     time=0,            // tutturma noktası zaman
                           double       price=0)           // tutturma noktası fiyat
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fibonacci Düzeltme Seviyelerini sil                              |
//+------------------------------------------------------------------+
bool FiboLevelsDelete(const long   chart_ID=0,        // çizelge tanımlayıcısı
                      const string name="FiboLevels") // nesne ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesneyi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Fibonacci Düzeltme Seviyeleri\" silinemedi! Hata kodu = ",GetLastErro
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fibonacci Düzeltme Seviyelerinin tutturma noktası değerlerini    |
//| varsayılan değerleri boş değerler için ayarla                    |
```

```
//+------------------------------------------------------------------+
void ChangeFiboLevelsEmptyPoints(datetime &time1,double &price1,
                                 datetime &time2,double &price2)
  {
//--- ikinci noktanın zamanı ayarlanmamışsa, mevcut çubuğun üstünde olacak
   if(!time2)
      time2=TimeCurrent();
//--- İkinci noktanın fiyatı ayarlanmamışsa, Bid (alış) değerini alacak
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- ilk noktanın zamanı ayarlanmamışsa, ikinci noktanın 9 çubuk solunda olacak
   if(!time1)
     {
      //--- son 10 çubuğun açılış zamanını alacak bir dizi
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- ilk noktayı ikinciden 9 çubuk sonraya ayarla
      time1=temp[0];
     }
//--- ilk noktanın fiyatı ayarlanmamışsa, ikincinin 200 puan altına yerleştir
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- Fibonacci Düzeltme Seviyelerinin tutturma noktalarını ayarlamak ve değiştirmek i
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
```

```
      {
       Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
       return;
      }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- Fibonacci Düzeltme Seviyelerini çizmek için noktaları tanımla
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- nesneyi oluştur
   if(!FiboLevelsCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidden,InpZOrde
      {
       return;
      }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, tutturma noktalarını taşı
//--- döngü sayacı
   int v_steps=accuracy*2/5;
//--- ilk tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
      {
       //--- bir sonraki değeri kullan
       if(p1>1)
          p1-=1;
       //--- tutturma noktasını taşı
       if(!FiboLevelsPointChange(0,InpName,0,date[d1],price[p1]))
          return;
       //--- script işlemi devre dışı bırakıldı mı kontrol et
       if(IsStopped())
          return;
       //--- çizelgeyi yenile
       ChartRedraw();
      }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   v_steps=accuracy*4/5;
//--- ikinci tutturma noktasını taşı
```

```
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p2>1)
         p2-=1;
      //--- tutturma noktasını taşı
      if(!FiboLevelsPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- nesneyi çizelgeden sil
   FiboLevelsDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_FIBOTIMES

Fibonacci Zaman Aralıkları.

Not

"Fibonacci Zaman Aralıkları" için, seviye çizgilerinin sayısını, rengini ve değerlerini belirlemek mümkündür.

Örnek

Aşağıdaki script, Fibonacci Zaman Aralıklarını çizelge üzerinde oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Fibonacci Zaman Aralıkları\" grafiksel nesnesini çize
#property description "Tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="FiboTimes";       // Nesne ismi
input int             InpDate1=10;               // 1-inci noktanın tarihi, %
input int             InpPrice1=45;              // 1-inci noktanın fiyatı, %
input int             InpDate2=20;               // 2-inci noktanın tarihi, %
input int             InpPrice2=55;              // 2-inci noktanın fiyatı, %
input color           InpColor=clrRed;           // Nesne rengi
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Çizgi stili
input int             InpWidth=2;                // Çizgi stili
```

```
input bool            InpBack=false;             // Arka-plan nesnesi
input bool            InpSelection=true;         // Taşımak için vurgula
input bool            InpHidden=true;            // Nesne listesinde gizle
input long            InpZOrder=0;               // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Fibonacci Zaman Aralıklarını verilen koordinatlarda oluştur      |
//+------------------------------------------------------------------+
bool FiboTimesCreate(const long            chart_ID=0,        // çizelge tanımlayıcı
                     const string          name="FiboTimes",  // nesne ismi
                     const int             sub_window=0,      // alt pencere indisi
                     datetime              time1=0,           // ilk nokta zamanı
                     double                price1=0,          // ilk nokta fiyatı
                     datetime              time2=0,           // ikinci nokta zamanı
                     double                price2=0,          // ikinci nokta fiyatı
                     const color           clr=clrRed,        // nesne rengi
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // nesne çizgi stili
                     const int             width=1,           // nesne çizgi kalınlığı
                     const bool            back=false,        // arka-plan çizimi
                     const bool            selection=true,    // taşıma için vurgula
                     const bool            hidden=true,       // nesne listesinde gizl
                     const long            z_order=0)         // fare tıklaması için ö
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeFiboTimesEmptyPoints(time1,price1,time2,price2);
//--- hata değerini sıfırla
   ResetLastError();
//--- verilen koordinatlarda Fibonacci zaman dilimlerini oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOTIMES,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": \"Fibonacci Zaman Aralıklarının\" oluşturulması başarısız oldu! Hata ko
      return(false);
     }
//--- rengi ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- çizgi genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

```
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Seviyelerin parametrelerini ve sayılarını ayarla                 |
//+------------------------------------------------------------------+
bool FiboTimesLevelsSet(int             levels,           // seviye çizgilerinin sayıs
                        double          &values[],        // seviye çizgilerinin değe
                        color           &colors[],        // seviye çizgilerinin rengi
                        ENUM_LINE_STYLE &styles[],        // seviye çizgilerinin stili
                        int             &widths[],        // seviye çizgilerinin kalın
                        const long      chart_ID=0,       // çizelge tanımlayıcı
                        const string    name="FiboTimes") // nesne ismi
  {
//--- dizi büyüklüklerini ayarla
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": dizi uzunluğu seviyelerin sayısıyla örtüşmüyor, hata!");
      return(false);
     }
//--- seviyelerin sayısını ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- seviyelerin özeliklerini döngü içinde ayarla
   for(int i=0;i<levels;i++)
     {
      //--- seviye değeri
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- seviye rengi
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- seviye stili
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- seviye genişliği
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- seviye açıklaması
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(values[i],1));
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fibonacci Zaman Aralıklarının tutturma noktası                   |
//+------------------------------------------------------------------+
bool FiboTimesPointChange(const long   chart_ID=0,       // çizelge tanımlayıcı
                          const string name="FiboTimes", // nesne ismi
                          const int    point_index=0,    // tutturma noktası indisi
                          datetime     time=0,           // tutturma noktası zaman koo
```

```
                           double       price=0)          // tutturma noktası fiyat koo
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fibonacci Zaman Aralıklarını sil                                 |
//+------------------------------------------------------------------+
bool FiboTimesDelete(const long   chart_ID=0,       // çizelge tanımlayıcı
                     const string name="FiboTimes") // nesne ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesneyi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Fibonacci Zaman Aralıkları\" silinemedi! Hata kodu = ",GetLastError()
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fibonacci Zaman Aralıklarının değerlerini kontrol et ve          |
//| boş olanlar için varsayılan değerler ile ayarla                  |
//+------------------------------------------------------------------+
void ChangeFiboTimesEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
//--- ilk noktanın zamanı ayarlanmamışsa mevcut çubuk üzerinde olacak
   if(!time1)
      time1=TimeCurrent();
//--- ilk noktanın fiyatı ayarlanmamışsa mevcut Bid değerini alacak
   if(!price1)
```

```
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- ikinci noktanın zamanı ayarlanmamışsa, ilk noktanın 2 çubuk solunda olacak
   if(!time2)
     {
      //--- son 3 çubuğun açılış zamanını almak için bir dizi
      datetime temp[3];
      CopyTime(Symbol(),Period(),time1,3,temp);
      //--- ilk noktayı ikincinin 2 çubuk soluna yerleştir
      time2=temp[0];
     }
//--- ikinci noktanın fiyatı ayarlanmamışsa birinciye eşit olur
   if(!price2)
      price2=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- Fibonacci Zaman Aralıklarının tutturma noktalarının koordinatlarını ayarlamak ve
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
```

```
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- Fibonacci Zaman Aralıklarını çizmek için noktaları tanımla
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- nesneyi oluştur
   if(!FiboTimesCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, tutturma noktalarını taşı
//--- döngü sayacı
   int h_steps=bars*2/5;
//--- ikinci tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d2<bars-1)
         d2+=1;
      //--- tutturma noktasını taşı
      if(!FiboTimesPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   h_steps=bars*3/5;
//--- ilk tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d1<bars-1)
         d1+=1;
      //--- tutturma noktasını taşı
      if(!FiboTimesPointChange(0,InpName,0,date[d1],price[p1]))
```

```
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- nesneyi çizelgeden sil
   FiboTimesDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_FIBOFAN

Fibonacci Yelpazesi.

Not

"Fibonacci Yelpazesi" için, seviye çizgilerinin sayılarının, değerlerinin ve renklerinin belirlenmesi mümkündür.

Örnek

Aşağıdaki script, Fibonacci Yelpazelerini çizelge üzerinde çizer ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Fibonacci Yelpazesi\" grafiksel nesnesini çizer."
#property description "Tutturma (çapa) noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="FiboFan";          // Yelpaze ismi
input int             InpDate1=10;                // 1-inci noktanın tarihi, %
input int             InpPrice1=25;               // 1-inci noktanın fiyatı, %
input int             InpDate2=30;                // 2-inci noktanın tarihi, %
input int             InpPrice2=50;               // 2-inci noktanın fiyatı, %
input color           InpColor=clrRed;            // Yelpaze çizgi rengi
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Çizgi stili
```

```
input int             InpWidth=2;                // Çizgi stili
input bool            InpBack=false;             // Arka-plan nesnesi
input bool            InpSelection=true;         // Taşımak için vurgula
input bool            InpHidden=true;            // Nesne listesinde gizle
input long            InpZOrder=0;               // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Fibonacci Yelpazesini verilen koordinatlarda oluştur             |
//+------------------------------------------------------------------+
bool FiboFanCreate(const long            chart_ID=0,        // çizelge tanımlayıcısı
                   const string          name="FiboFan",    // yelpaze ismi
                   const int             sub_window=0,      // alt-pencere indisi
                   datetime              time1=0,           // ilk noktanın zamanı
                   double                price1=0,          // ilk noktanın fiyatı
                   datetime              time2=0,           // ikinci nokta zamanı
                   double                price2=0,          // ikinci nokta fiyatı
                   const color           clr=clrRed,        // yelpaze çizgisi rengi
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // yelpaze çizgisi rengi
                   const int             width=1,           // yelpaze çizgisi kalınlı
                   const bool            back=false,        // arka-planda
                   const bool            selection=true,    // taşıma için vurgula
                   const bool            hidden=true,       // nesne listesinde gizle
                   const long            z_order=0)         // fare tıklaması için önc
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeFiboFanEmptyPoints(time1,price1,time2,price2);
//--- hata değerini sıfırla
   ResetLastError();
//--- Fibonacci Yelpazesini verilen koordinatlarda oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOFAN,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": \"Fibonacci Yelpazesinin\" oluşturulması başarısız! Hata kodu = ",GetLa
      return(false);
     }
//--- rengi ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- çizgi genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Seviyelerin parametrelerini ve sayılarını ayarla                 |
//+------------------------------------------------------------------+
bool FiboFanLevelsSet(int              levels,         // seviye çizgilerinin sayısı
                      double           &values[],      // seviye çizgilerinin değeri
                      color            &colors[],      // seviye çizgilerinin rengi
                      ENUM_LINE_STYLE &styles[],       // seviye çizgilerinin stili
                      int              &widths[],      // seviye çizgilerinin kalınlığı
                      const long       chart_ID=0,     // çizelge kimliği
                      const string     name="FiboFan") // yelpaze ismi
  {
//--- dizi büyüklüklerini ayarla
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": dizi uzunluğu seviyelerin sayısıyla örtüşmüyor, hata!");
      return(false);
     }
//--- seviyelerin sayısını ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- seviyelerin özeliklerini döngü içinde ayarla
   for(int i=0;i<levels;i++)
     {
      //--- seviye değeri
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- seviye rengi
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- seviye stili
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- seviye genişliği
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- seviye açıklaması
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fibonacci Yelpazesi tutturma noktasını taşı                      |
//+------------------------------------------------------------------+
bool FiboFanPointChange(const long   chart_ID=0,     // çizelge kimliği
                        const string name="FiboFan", // yelpaze ismi
                        const int    point_index=0,  // tutturma noktası indisi
```

```
                        datetime     time=0,         // tutturma noktası zaman koordi
                        double       price=0)        // tutturma noktası fiyat koordi
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğ
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Yelpazeyi sil                                                    |
//+------------------------------------------------------------------+
bool FiboFanDelete(const long   chart_ID=0,     // çizelge tanımlayıcısı
                   const string name="FiboFan") // yelpaze ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- yelpazeyi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Fibonacci Yelpazesinin\" silinmesi başarısız oldu! Hata kodu = ",GetL
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fibonacci Yelpazesi tutturma noktası değerlerini kontrol et ve   |
//| varsayılan değerleri boş değerler için ayarla                    |
//+------------------------------------------------------------------+
void ChangeFiboFanEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- ikinci noktanın zamanı ayarlanmamışsa, mevcut çubuğun üstünde olacak
   if(!time2)
      time2=TimeCurrent();
//--- İkinci noktanın fiyatı ayarlanmamışsa, Bid (alış) değerini alacak
```

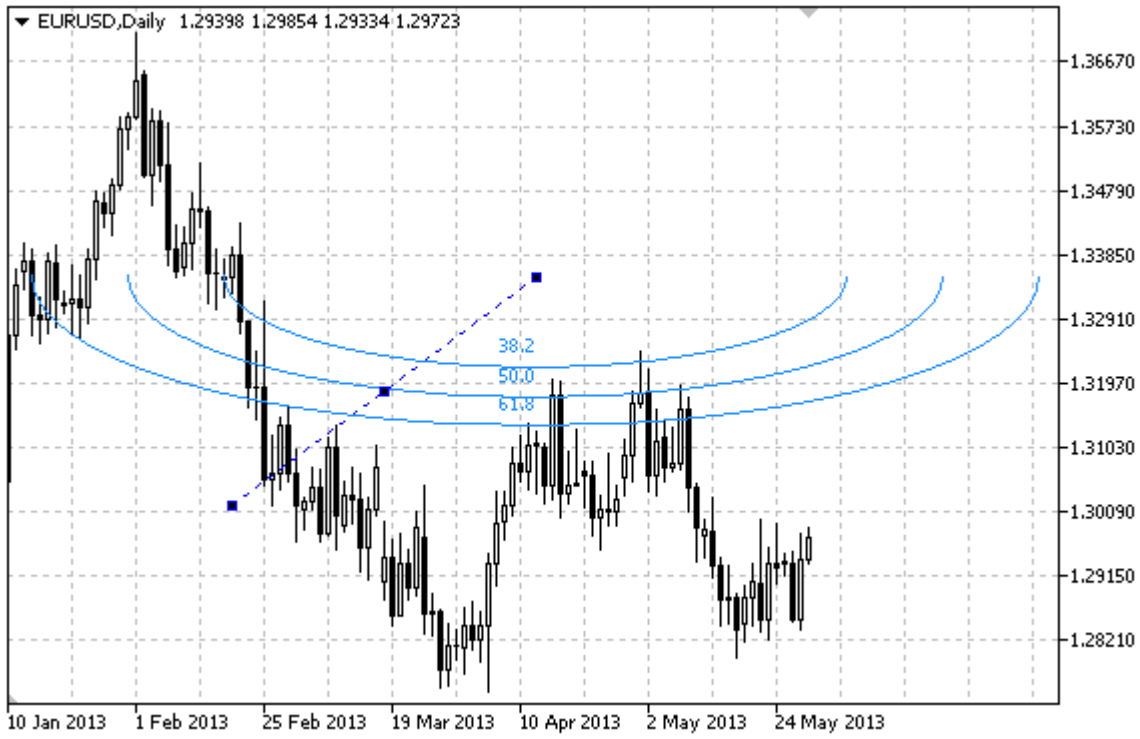
```
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- ilk noktanın zamanı ayarlanmamışsa, ikinci noktanın 9 çubuk solunda olacak
   if(!time1)
     {
      //--- son 10 çubuğun açılış zamanını alacak bir dizi
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- ilk noktayı ikinciden 9 çubuk sonraya ayarla
      time1=temp[0];
     }
//--- ilk noktanın fiyatı ayarlanmamışsa, ikincinin 200 puan altına yerleştir
   if(!price1)
      price1=price2-200*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- Fibonacci Yelpazesinin tutturma noktası koordinatlarını ayarlamak ve değiştirmek
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
```

```
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- Fibonacci Yelpazesinin çizileceği noktalar
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- nesneyi oluştur
   if(!FiboFanCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- yelpazenin tutturma noktalarını taşı
//--- döngü sayacı
   int v_steps=accuracy/2;
//--- ilk tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p1<accuracy-1)
         p1+=1;
      //--- tutturma noktasını taşı
      if(!FiboFanPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   int h_steps=bars/4;
//--- ikinci tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d2<bars-1)
         d2+=1;
      //--- tutturma noktasını taşı
      if(!FiboFanPointChange(0,InpName,1,date[d2],price[p2]))
         return;
```

```
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- nesneyi çizelgeden sil
   FiboFanDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_FIBOARC

Fibonacci Yayları.



Not

"Fibonacci Yayları" için, bütün bir elipsin görüntülenmesi mümkündür. Eğrisel yarıçap, tutturma noktasının ölçeğini ve koordinatlarını değiştirerek belirlenebilir.

Ayrıca, seviye çizgilerinin sayısını, değerlerini ve renklerini de belirtebilirsiniz.

Örnek

Aşağıdaki script, çizelge üzerinde Fibonacci Yaylarını oluşturur. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script \"Fibonacci Yayları\" grafiksel nesnesini çizer."
#property description "Tutturma (çapa) noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="FiboArc";         // Nesne ismi
input int             InpDate1=25;               // 1-inci noktanın tarihi, %
input int             InpPrice1=25;              // 1-inci noktanın fiyatı, %
input int             InpDate2=35;               // 2-inci noktanın tarihi, %
input int             InpPrice2=55;              // 2-inci noktanın fiyatı, %
input double          InpScale=3.0;              // Ölçek
```

```
input bool             InpFullEllipse=true;       // Yay şekli
input color            InpColor=clrRed;           // Çizgi rengi
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Çizgi stili
input int              InpWidth=2;                // Çizgi stili
input bool             InpBack=false;             // Arka-plan nesnesi
input bool             InpSelection=true;         // Taşımak için vurgula
input bool             InpHidden=true;            // Nesne listesinde gizle
input long             InpZOrder=0;               // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Verilen koordinatlarda Fibonacci Yaylarını oluştur               |
//+------------------------------------------------------------------+
bool FiboArcCreate(const long             chart_ID=0,          // çizelge kimliği
                   const string           name="FiboArc",      // nesne ismi
                   const int              sub_window=0,        // alt-pencere indisi
                   datetime               time1=0,             // ilk noktanın zamanı
                   double                 price1=0,            // ilk noktanın fiyatı
                   datetime               time2=0,             // ikinci noktanın zamanı
                   double                 price2=0,            // ikinci noktanın fiyatı
                   const double           scale=1.0,           // ölçek
                   const bool             full_ellipse=false,  // yay şekli
                   const color            clr=clrRed,          // çizgi rengi
                   const ENUM_LINE_STYLE style=STYLE_SOLID,   // çizgi stili
                   const int              width=1,             // çizgi kalınlığı
                   const bool             back=false,          // arka-planda
                   const bool             selection=true,      // taşıma için vurgula
                   const bool             hidden=true,         // nesne listesinde gizle
                   const long             z_order=0)           // fare tıklaması için ön
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeFiboArcEmptyPoints(time1,price1,time2,price2);
//--- hata değerini sıfırla
   ResetLastError();
//--- Verilen koordinatlarda Fibonacci Yaylarını oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOARC,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": \"Fibonacci Yaylarının\" oluşturulması başarısız! Hata kodu = ",GetLast
      return(false);
     }
//--- ölçeği ayarla
   ObjectSetDouble(chart_ID,name,OBJPROP_SCALE,scale);
//--- yayların görünümünü bütün (true) veya yarım (false) elips şeklinde görüntüle
   ObjectSetInteger(chart_ID,name,OBJPROP_ELLIPSE,full_ellipse);
//--- rengi ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- çizgi genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

```
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Seviyelerin parametrelerini ve sayılarını ayarla                 |
//+------------------------------------------------------------------+
bool FiboArcLevelsSet(int              levels,         // seviye çizgilerinin sayısı
                      double           &values[],      // seviye çizgilerinin değeri
                      color            &colors[],      // seviye çizgilerinin rengi
                      ENUM_LINE_STYLE  &styles[],      // seviye çizgilerinin stili
                      int              &widths[],      // seviye çizgilerinin kalınlığı
                      const long       chart_ID=0,     // çizelge kimliği
                      const string     name="FiboArc") // nesne ismi
  {
//--- dizi büyüklüklerini ayarla
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": dizi uzunluğu seviyelerin sayısıyla örtüşmüyor, hata!");
      return(false);
     }
//--- seviyelerin sayısını ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- seviyelerin özeliklerini döngü içinde ayarla
   for(int i=0;i<levels;i++)
     {
      //--- seviye değeri
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- seviye rengi
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- seviye stili
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- seviye genişliği
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- seviye açıklaması
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
```

```
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fibonacci Yaylarının tutturma noktasını taşı                     |
//+------------------------------------------------------------------+
bool FiboArcPointChange(const long   chart_ID=0,     // çizelge kimliği
                        const string name="FiboArc", // nesne ismi
                        const int    point_index=0,  // tutturma noktası indisi
                        datetime     time=0,         // tutturma noktası zaman koordin
                        double       price=0)        // tutturma noktası fiyat koordin
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fibonacci Yaylarını sil                                          |
//+------------------------------------------------------------------+
bool FiboArcDelete(const long   chart_ID=0,     // çizelge kimliği
                   const string name="FiboArc") // nesne ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesneyi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Fibonacci Yaylarının\" silinmesi başarısız oldu! Hata kodu = ",GetLas
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fibonacci Yaylarının tutturma noktası değerlerini kontrol et ve  |
```

```
//| boş olanlar için varsayılan değerleri kullan                     |
//+------------------------------------------------------------------+
void ChangeFiboArcEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2)
  {
//--- ikinci noktanın zamanı ayarlanmamışsa, mevcut çubuğun üstünde olacak
   if(!time2)
      time2=TimeCurrent();
//--- İkinci noktanın fiyatı ayarlanmamışsa, Bid (alış) değerini alacak
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- ilk noktanın zamanı ayarlanmamışsa, ikinci noktanın 9 çubuk solunda olacak
   if(!time1)
     {
      //--- son 10 çubuğun açılış zamanını alacak bir dizi
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- ilk noktayı ikinciden 9 çubuk sonraya ayarla
      time1=temp[0];
     }
//--- ilk noktanın fiyatı ayarlanmamışsa, ikincinin 300 puan aşağısına yerleştir
   if(!price1)
      price1=price2-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- Fibonacci Yaylarının tutturma noktası koordinatlarını ayarlamak ve değiştirmek i
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
```

```
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- Fibonacci Yaylarını çizmek için noktaları tanımla
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- nesneyi oluştur
   if(!FiboArcCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpScale,
      InpFullEllipse,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrd
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, tutturma noktalarını taşı
//--- döngü sayacı
   int v_steps=accuracy/5;
//--- ilk tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p1<accuracy-1)
         p1+=1;
      //--- tutturma noktasını taşı
      if(!FiboArcPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   int h_steps=bars/5;
```

```
//--- ikinci tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d2<bars-1)
         d2+=1;
      //--- tutturma noktasını taşı
      if(!FiboArcPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- nesneyi çizelgeden sil
   FiboArcDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_FIBOCHANNEL

Fibonacci Kanalı.

Not

Fibonacci Kanalının, sağa ve/veya sola doğru olan sürekliliklerinin (sırasıyla OBJPROP_RAY_RIGHT ve OBJPROP_RAY_LEFT özelliklerinin) belirtilmesi mümkündür.

Ayrıca, seviye çizgilerinin sayısını, değerlerini ve renklerini de belirtebilirsiniz.

Örnek

Aşağıdaki script, çizelge üzerinde Fibonacci Kanallarını oluşturur. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Fibonacci Kanalı\" grafiksel nesnesini çizer."
#property description "Tutturma (çapa) noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="FiboChannel";     // Kanal ismi
input int             InpDate1=20;               // 1-inci noktanın tarihi, %
input int             InpPrice1=10;              // 1-inci noktanın fiyatı, %
input int             InpDate2=60;               // 2-inci noktanın tarihi, %
input int             InpPrice2=30;              // 2-inci noktanın fiyatı, %
input int             InpDate3=20;               // 3-üncü noktanın tarihi, %
```

```
input int              InpPrice3=25;              // 3-üncü noktanın fiyatı, %
input color            InpColor=clrRed;           // Kanal rengi
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // Kanal çizgilerinin stili
input int              InpWidth=2;                // Kanal çizgilerinin genişliği
input bool             InpBack=false;             // Arkaplan kanalı
input bool             InpSelection=true;         // Taşımak için vurgula
input bool             InpRayLeft=false;          // Kanalın sola sürekliliği
input bool             InpRayRight=false;         // Kanalın sağa sürekliliği
input bool             InpHidden=true;            // Nesne listesinde gizle
input long             InpZOrder=0;               // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Verilen koordinatlarda Fibonacci Kanalını oluştur                |
//+------------------------------------------------------------------+
bool FiboChannelCreate(const long             chart_ID=0,         // çizelge kimliği
                       const string           name="FiboChannel", // kanal ismi
                       const int              sub_window=0,       // alt-pencere indisi
                       datetime               time1=0,            // ilk noktanın zaman
                       double                 price1=0,           // ilk noktanın fiyat
                       datetime               time2=0,            // ikinci noktanın za
                       double                 price2=0,           // ikinci noktanın fi
                       datetime               time3=0,            // üçüncü nokta zaman
                       double                 price3=0,           // üçüncü nokta fiyat
                       const color            clr=clrRed,         // kanal rengi
                       const ENUM_LINE_STYLE  style=STYLE_SOLID,  // kanal çizgilerinin
                       const int              width=1,            // kanal çizgilerinin
                       const bool             back=false,         // arka-planda
                       const bool             selection=true,     // taşımak için vurgu
                       const bool             ray_left=false,     // kanalın sola doğru
                       const bool             ray_right=false,    // kanalın sağa doğru
                       const bool             hidden=true,        // nesne listesinde s
                       const long             z_order=0)          // fare tıklaması içi
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeFiboChannelEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- hata değerini sıfırla
   ResetLastError();
//--- verilen koordinatlarda bir kanal oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_FIBOCHANNEL,sub_window,time1,price1,time2,price2
     {
      Print(__FUNCTION__,
            ": \"Fibonacci Kanalı\" oluşturulamadı! Hata kodu = ",GetLastError());
      return(false);
     }
//--- kanal rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- kanal çizgilerinin stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- kanal çizgilerinin genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

```
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- kanalın sola doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı k
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- kanalın sağa doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı k
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Seviyelerin parametrelerini ve sayılarını ayarla                 |
//+------------------------------------------------------------------+
bool FiboChannelLevelsSet(int             levels,             // seviye çizgilerinin s
                          double          &values[],          // seviye çizgilerinin d
                          color           &colors[],          // seviye çizgilerinin r
                          ENUM_LINE_STYLE &styles[],          // seviye çizgilerinin s
                          int             &widths[],          // seviye çizgilerinin k
                          const long      chart_ID=0,         // çizelge kimliği
                          const string    name="FiboChannel") // nesne ismi
  {
//--- dizi büyüklüklerini ayarla
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": dizi uzunluğu seviyelerin sayısıyla örtüşmüyor, hata!");
      return(false);
     }
//--- seviyelerin sayısını ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- seviyelerin özeliklerini döngü içinde ayarla
   for(int i=0;i<levels;i++)
     {
      //--- seviye değeri
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- seviye rengi
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- seviye stili
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- seviye genişliği
```

```
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- seviye açıklaması
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,DoubleToString(100*values[i],1
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fibonacci Kanalının tutturma noktasını taşı                      |
//+------------------------------------------------------------------+
bool FiboChannelPointChange(const long   chart_ID=0,         // çizelge tanımlayıcısı
                            const string name="FiboChannel", // kanal ismi
                            const int    point_index=0,      // tutturma (çapa) noktas
                            datetime     time=0,             // tutturma noktası zaman
                            double       price=0)            // tutturma noktası fiyat
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Kanalı sil                                                       |
//+------------------------------------------------------------------+
bool FiboChannelDelete(const long   chart_ID=0,         // çizelge kimliği
                       const string name="FiboChannel") // kanal ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- kanalı sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Fibonacci Kanalı\" silinemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
```

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Fibonacci Kanalının tutturma noktası değerlerini kontrol et ve   |
//| varsayılan değerleri boş değerler için ayarla                    |
//+------------------------------------------------------------------+
void ChangeFiboChannelEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                  double &price2,datetime &time3,double &price3)
  {
//--- İkinci noktanın (sağ) zamanı ayarlanmamışsa, ilk çubuk üzerinde olacak
   if(!time2)
      time2=TimeCurrent();
//--- İkinci noktanın fiyatı ayarlanmamışsa, Bid (alış) değerini alacak
   if(!price2)
      price2=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- İlk noktanın (sol) zamanı ayarlanmamışsa, ikinciden 9 çubuk sonra olacak
   if(!time1)
     {
      //--- son 10 çubuğun açılış zamanını alacak bir dizi
      datetime temp[10];
      CopyTime(Symbol(),Period(),time2,10,temp);
      //--- ilk noktayı ikinciden 9 çubuk sonraya ayarla
      time1=temp[0];
     }
//--- ilk noktanın fiyatı ayarlanmamışsa, ikincinin 300 puan üstüne taşı
   if(!price1)
      price1=price2+300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- üçüncü noktanın zamanı ayarlanmamışsa, birinci noktanın üstüne gelsin
   if(!time3)
      time3=time1;
//--- üçüncü noktanın fiyatı ayarlanmamışsa, ikinci noktanın fiyatına eşittir
   if(!price3)
      price3=price2;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   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("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

```
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- kanalın çizileceği noktaları tanımla
   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;
//--- Fibonacci Kanalını oluştur
   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;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, kanalın tutturma noktalarını taşı
//--- döngü sayacı
   int h_steps=bars/10;
//--- ilk tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d1>1)
         d1-=1;
```

```
      //--- tutturma noktasını taşı
      if(!FiboChannelPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   int v_steps=accuracy/10;
//--- ikinci tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p2>1)
         p2-=1;
      //--- tutturma noktasını taşı
      if(!FiboChannelPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   v_steps=accuracy/15;
//--- üçüncü tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p3<accuracy-1)
         p3+=1;
      //--- tutturma noktasını taşı
      if(!FiboChannelPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
```

```
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- kanalı çizelgeden sil
   FiboChannelDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_EXPANSION

Fibonacci Açılımı.

Not

Bir "Fibonacci Açılımının" sağa ve/veya sola doğru olan sürekliliklerinin (sırasıyla OBJPROP_RAY_RIGHT ve OBJPROP_RAY_LEFT özelliklerinin) belirtilmesi mümkündür.

Ayrıca, seviye çizgilerinin sayısını, değerlerini ve renklerini de belirtebilirsiniz.

Örnek

Aşağıdaki script, çizelge üzerinde Fibonacci Açılımı nesnesini oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script \"Fibonacci Açılımı\" grafiksel nesnesini çizer."
#property description "Tutturma (çapa) noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="FiboExpansion";   // Nesne ismi
input int             InpDate1=10;               // 1-inci noktanın tarihi, %
input int             InpPrice1=55;              // 1-inci noktanın fiyatı, %
input int             InpDate2=30;               // 2-inci noktanın tarihi, %
input int             InpPrice2=10;              // 2-inci noktanın fiyatı, %
input int             InpDate3=80;               // 3-üncü noktanın tarihi, %
```

```
input int              InpPrice3=75;              // 3-üncü noktanın fiyatı, %
input color            InpColor=clrRed;           // Nesne rengi
input ENUM_LINE_STYLE  InpStyle=STYLE_DASHDOTDOT; // Çizgi stili
input int              InpWidth=2;                // Çizgi kalınlığı
input bool             InpBack=false;             // Arka-plan nesnesi
input bool             InpSelection=true;         // Taşımak için vurgula
input bool             InpRayLeft=false;          // Nesnenin sola sürekliliği
input bool             InpRayRight=false;         // Nesnenin sağa sürekliliği
input bool             InpHidden=true;            // Nesne listesinde gizle
input long             InpZOrder=0;               // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Verilen koordinatlarla Fibonacci Açılımını Oluştur               |
//+------------------------------------------------------------------+
bool FiboExpansionCreate(const long             chart_ID=0,           // çizelge kimliğ
                         const string           name="FiboExpansion", // kanal ismi
                         const int              sub_window=0,         // alt-pencere in
                         datetime               time1=0,              // ilk nokta zama
                         double                 price1=0,             // ilk nokta fiya
                         datetime               time2=0,              // ikinci nokta z
                         double                 price2=0,             // ikinci nokta f
                         datetime               time3=0,              // üçüncü nokta z
                         double                 price3=0,             // üçüncü nokta f
                         const color            clr=clrRed,           // nesne rengi
                         const ENUM_LINE_STYLE  style=STYLE_SOLID,    // çizgi stili
                         const int              width=1,              // çizgi kalınlığ
                         const bool             back=false,           // arka-plan nesn
                         const bool             selection=true,       // taşıma için vu
                         const bool             ray_left=false,       // nesnenin sola
                         const bool             ray_right=false,      // nesnenin sağa
                         const bool             hidden=true,          // nesne listesin
                         const long             z_order=0)            // fare tıklaması
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeFiboExpansionEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- hata değerini sıfırla
   ResetLastError();
//--- Fibonacci Açılımını verilen koordinatlarda oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_EXPANSION,sub_window,time1,price1,time2,price2,t
     {
      Print(__FUNCTION__,
            ": \"Fibonacci Açılımını\" oluşturma başarısız! Hata kodu = ",GetLastError
      return(false);
     }
//--- nesnenin rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- çizgilerin genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
```

```
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesnenin sola doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_LEFT,ray_left);
//--- nesnenin sağa doğru sürekli gösterimi modunu etkinleştir (true) veya devre dışı
   ObjectSetInteger(chart_ID,name,OBJPROP_RAY_RIGHT,ray_right);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Seviyelerin parametrelerini ve sayılarını ayarla                 |
//+------------------------------------------------------------------+
bool FiboExpansionLevelsSet(int             levels,                // seviye çizgileri
                            double          &values[],             // seviye çizgileri
                            color           &colors[],             // seviye çizgileri
                            ENUM_LINE_STYLE &styles[],             // seviye çizgileri
                            int             &widths[],             // seviye çizgileri
                            const long      chart_ID=0,            // çizelge kimliği
                            const string    name="FiboExpansion") // nesne ismi
  {
//--- dizi büyüklüklerini ayarla
   if(levels!=ArraySize(colors) || levels!=ArraySize(styles) ||
      levels!=ArraySize(widths) || levels!=ArraySize(widths))
     {
      Print(__FUNCTION__,": dizi uzunluğu seviyelerin sayısıyla örtüşmüyor, hata!");
      return(false);
     }
//--- seviyelerin sayısını ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_LEVELS,levels);
//--- seviyelerin özeliklerini döngü içinde ayarla
   for(int i=0;i<levels;i++)
     {
      //--- seviye değeri
      ObjectSetDouble(chart_ID,name,OBJPROP_LEVELVALUE,i,values[i]);
      //--- seviye rengi
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELCOLOR,i,colors[i]);
      //--- seviye stili
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELSTYLE,i,styles[i]);
      //--- seviye genişliği
```

```
      ObjectSetInteger(chart_ID,name,OBJPROP_LEVELWIDTH,i,widths[i]);
      //--- seviye açıklaması
      ObjectSetString(chart_ID,name,OBJPROP_LEVELTEXT,i,"FE "+DoubleToString(100*value
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fibonacci Açılımının tutturma noktasını taşı                     |
//+------------------------------------------------------------------+
bool FiboExpansionPointChange(const long   chart_ID=0,           // çizelge kimliği
                              const string name="FiboExpansion", // nesne ismi
                              const int    point_index=0,        // tutturma noktası
                              datetime     time=0,               // tutturma noktası
                              double       price=0)              // tutturma noktası
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Fibonacci Açılımını sil                                          |
//+------------------------------------------------------------------+
bool FiboExpansionDelete(const long   chart_ID=0,           // çizelge tanımlayıcısı
                         const string name="FiboExpansion") // nesne ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesneyi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Fibonacci Açılımı\" nesnesi silinemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
```

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Fibonacci Açılımının tutturma noktası değerlerini kontrol et ve  |
//| varsayılan değerleri boş değerler için ayarla                    |
//+------------------------------------------------------------------+
void ChangeFiboExpansionEmptyPoints(datetime &time1,double &price1,datetime &time2,
                                    double &price2,datetime &time3,double &price3)
  {
//--- üçüncü (right) noktanın zamanı ayarlanmamışsa, mevcut çubuk üzerinde olacak
   if(!time3)
      time3=TimeCurrent();
//--- üçüncü (sağ) noktanın fiyatı ayarlanmamışsa, Bid değerini alacak
   if(!price3)
      price3=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- ilk (sol) noktanın zamanı ayarlanmamışsa, üçüncüden 9 çubuk sonraya konumlanacak
//--- son 10 çubuğun açılış zamanlarını almak için bir dizi
   datetime temp[];
   ArrayResize(temp,10);
   if(!time1)
     {
      CopyTime(Symbol(),Period(),time3,10,temp);
      //--- ilk noktayı ikinciden 9 çubuk sonraya ayarla
      time1=temp[0];
     }
//--- ilk noktanın fiyatı ayarlanmamışsa, üçüncü noktanınkine eşit olacak
   if(!price1)
      price1=price3;
//--- ikinci noktanın zamanı ayarlanmamışsa, üçüncünün 7 çubuk soluna konumlanır
   if(!time2)
      time2=temp[2];
//--- ikinci noktanın fiyatı ayarlanmamışsa, ilkinden 250 puan aşağıya taşı
   if(!price2)
      price2=price1-250*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   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("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
```

```
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- Fibonacci Açılımı için noktaları tanımla
   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;
//--- Fibonacci Açılımını oluştur
   if(!FiboExpansionCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],
      InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpRayLeft,InpRayRight,InpHidder
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, tutturma noktalarını taşı
//--- döngü sayacı
   int v_steps=accuracy/10;
//--- ilk tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p1>1)
```

```
         p1-=1;
      //--- tutturma noktasını taşı
      if(!FiboExpansionPointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   v_steps=accuracy/2;
//--- üçüncü tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p3>1)
         p3-=1;
      //--- tutturma noktasını taşı
      if(!FiboExpansionPointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   v_steps=accuracy*4/5;
//--- ikinci tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p2<accuracy-1)
         p2+=1;
      //--- tutturma noktasını taşı
      if(!FiboExpansionPointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
```

```
   Sleep(1000);
//--- nesneyi çizelgeden sil
   FiboExpansionDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_ELLIOTWAVE5

Elliott Dürtü Dalgası.

Not

"Elliott Dürtü Dalgası" için, noktaların çizgilerle bağlanması modunun (OBJPROP_DRAWLINES özelliğinin), etkinleştirilmesi/devre dışı bırakılması mümkündür, aynı zamanda dalganın konumlandırılması da (ENUM_ELLIOT_WAVE_DEGREE sayımından) mümkündür.

Örnek

Aşağıdaki script, Elliott Dürtü Dalgasını çizelge üzerinde oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Elliott Dürtü Dalgası\" nesnesini çizer."
#property description "Tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string                  InpName="ElliotWave5";   // Nesne ismi
input int                     InpDate1=10;             // 1-inci noktanın tarihi, %
input int                     InpPrice1=90;            // 1-inci noktanın fiyatı, %
input int                     InpDate2=20;             // 2-inci noktanın tarihi, %
input int                     InpPrice2=40;            // 2-inci noktanın fiyatı, %
input int                     InpDate3=30;             // 3-üncü noktanın tarihi, %
```

```
input int                      InpPrice3=60;            // 3-üncü noktanın fiyatı, %
input int                      InpDate4=40;             // 4-üncü noktanın tarihi, %
input int                      InpPrice4=10;            // 4-üncü noktanın fiyatı, %
input int                      InpDate5=60;             // 5-inci noktanın tarihi, %
input int                      InpPrice5=40;            // 5-inci noktanın fiyatı, %
input ENUM_ELLIOT_WAVE_DEGREE InpDegree=ELLIOTT_MINOR; // Seviye
input bool                     InpDrawLines=true;       // Çizgilerin görüntülenmesi
input color                    InpColor=clrRed;         // Çizgilerin rengi
input ENUM_LINE_STYLE          InpStyle=STYLE_DASH;     // Çizgilerin stili
input int                      InpWidth=2;              // Çizgilerin kalınlığı
input bool                     InpBack=false;           // Arka-plan nesnesi
input bool                     InpSelection=true;       // Taşıma için vurgula
input bool                     InpHidden=true;          // Nesne listesinde gizle
input long                     InpZOrder=0;             // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Elliott Dürtü Dalgasını verilen koordinatlarda oluştur           |
//+------------------------------------------------------------------+
bool ElliotWave5Create(const long                  chart_ID=0,              // çize
                       const string                name="ElliotWave5",      // dalga
                       const int                   sub_window=0,            // alt-p
                       datetime                    time1=0,                 // ilk n
                       double                      price1=0,                // ilk n
                       datetime                    time2=0,                 // ikinc
                       double                      price2=0,                // ikinc
                       datetime                    time3=0,                 // üçünc
                       double                      price3=0,                // üçünc
                       datetime                    time4=0,                 // dördü
                       double                      price4=0,                // dördü
                       datetime                    time5=0,                 // beşin
                       double                      price5=0,                // beşin
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // derec
                       const bool                  draw_lines=true,         // çizgi
                       const color                 clr=clrRed,              // nesne
                       const ENUM_LINE_STYLE       style=STYLE_SOLID,       // çizgi
                       const int                   width=1,                 // çizgi
                       const bool                  back=false,              // arka-
                       const bool                  selection=true,          // taşın
                       const bool                  hidden=true,             // nesne
                       const long                  z_order=0)               // fare
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeElliotWave5EmptyPoints(time1,price1,time2,price2,time3,price3,time4,price4,ti
//--- hata değerini sıfırla
   ResetLastError();
//--- "Elliott Dürtü Dalgasını" verilen koordinatlarda oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE5,sub_window,time1,price1,time2,price2
      price3,time4,price4,time5,price5))
     {
      Print(__FUNCTION__,
```

```
            ": \"Elliott Dürtü Dalgası\" nesnesinin oluşturulması başarısız! Hata kodu
      return(false);
     }
//--- dereceyi (dalga boyunu) ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- çizgilerin görüntülenmesi modunu etkinleştir (true) veya devre dışı bırak (false
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- nesnenin rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- çizgilerin genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Elliott Dürtü Dalgasının tutturma noktasını taşı                 |
//+------------------------------------------------------------------+
bool ElliotWave5PointChange(const long   chart_ID=0,          // çizelge kimliği
                            const string name="ElliotWave5", // nesne ismi
                            const int    point_index=0,      // tutturma (çapa) noktas
                            datetime     time=0,             // tutturma noktası zaman
                            double       price=0)            // tutturma noktası fiyat
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
```

```
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Elliott Dürtü Dalgasını sil                                      |
//+------------------------------------------------------------------+
bool ElliotWave5Delete(const long   chart_ID=0,         // çizelge kimliği
                       const string name="ElliotWave5") // nesne ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesneyi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Elliott Dürtü Dalgasının\" silinmesi başarısız! Hata kodu = ",GetLast
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Elliott Dürtü Dalgasının tutturma noktalarının değerlerini       |
//| boş olanlar için varsayılan değerler ile ayarla                  |
//+------------------------------------------------------------------+
void ChangeElliotWave5EmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2,
                                  datetime &time3,double &price3,
                                  datetime &time4,double &price4,
                                  datetime &time5,double &price5)
  {
//--- son 10 çubuğun açılış zamanlarını almak için bir dizi
   datetime temp[];
   ArrayResize(temp,10);
//--- veriyi al
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- çizelge üzerindeki bir noktanın değerini al
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- ilk noktanın zamanı ayarlanmamışsa, son çubuktan 9 çubuk solda olacak
   if(!time1)
      time1=temp[0];
//--- ilk noktanın fiyatı ayarlanmamışsa mevcut Bid değerini alacak
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- ikinci noktanın zamanı ayarlı değilse, son çubuktan 7 soldaki çubukta olacak
   if(!time2)
      time2=temp[2];
```

```
//--- ikinci noktanın fiyatı ayarlanmamışsa, ilkinin 300 puan aşağısına konumla
   if(!price2)
      price2=price1-300*point;
//--- üçüncü noktanın zamanı ayarlanmamışsa, son çubuktan 5 çubuk solda olacak
   if(!time3)
      time3=temp[4];
//--- üçüncü noktanın fiyatı ayarlanmamışsa, ilkinden 250 puan aşağı konumla
   if(!price3)
      price3=price1-250*point;
//--- dördüncü noktanın zamanı ayarlanmamışsa, son çubuktan 3 çubuk solda olacak
   if(!time4)
      time4=temp[6];
//--- dördüncü noktanın fiyatı ayarlanmamışsa, ilkinden 550 puan aşağıda konumla
   if(!price4)
      price4=price1-550*point;
//--- beşinci noktanın zamanı ayarlanmamışsa, son çubuğun üstünde olacak
   if(!time5)
      time5=temp[9];
//--- beşinci noktanın fiyatı ayarlanmamışsa, ilkinden 450 puan aşağıda konumla
   if(!price5)
      price5=price1-450*point;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   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("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
```

```
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- Elliott Dürtü Dalgasının çizimi için noktaları tanımla
   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;
//--- Elliott Dürtü Dalgasını oluştur
   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;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, tutturma noktalarını taşı
//--- döngü sayacı
   int v_steps=accuracy/5;
//--- beşinci tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p5<accuracy-1)
         p5+=1;
      //--- tutturma noktasını taşı
      if(!ElliotWave5PointChange(0,InpName,4,date[d5],price[p5]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
```

```
         return;
       //--- çizelgeyi yenile
       ChartRedraw();
      }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   v_steps=accuracy/5;
//--- ikinci ve üçüncü tutturma noktalarını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- şu değerleri kullan
      if(p2<accuracy-1)
         p2+=1;
      if(p3>1)
         p3-=1;
      //--- noktaları kaydır
      if(!ElliotWave5PointChange(0,InpName,1,date[d2],price[p2]))
         return;
      if(!ElliotWave5PointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   v_steps=accuracy*4/5;
//--- birinci ve dördüncü tutturma (çapa) noktalarını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- şu değerleri kullan
      if(p1>1)
         p1-=1;
      if(p4<accuracy-1)
         p4+=1;
      //--- noktaları kaydır
      if(!ElliotWave5PointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ElliotWave5PointChange(0,InpName,3,date[d4],price[p4]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
```

```
      }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- nesneyi çizelgeden sil
   ElliotWave5Delete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_ELLIOTWAVE3

Elliott Düzeltme Dalgası.

Not

"Elliott Düzeltme Dalgası" için, noktaların çizgilerle bağlanması modunun (OBJPROP_DRAWLINES özelliğinin), etkinleştirilmesi/ devre dışı bırakılması mümkündür, aynı zamanda dalganın konumlandırılması da (ENUM_ELLIOT_WAVE_DEGREE sayımından) mümkündür.

Örnek

Aşağıdaki script, Elliott Düzeltme Dalgasını çizelge üzerinde oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Elliott Düzeltme Dalgası\" grafiksel nesnesini çizer.
#property description "Tutturma noktasının koordinatları, çizelge boyutunun"
#property description "yüzdesi olarak ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string                InpName="ElliotWave3";   // Nesne ismi
input int                   InpDate1=10;             // 1-inci noktanın tarihi, %
input int                   InpPrice1=90;            // 1-inci noktanın fiyatı, %
input int                   InpDate2=30;             // 2-inci noktanın tarihi, %
input int                   InpPrice2=10;            // 2-inci noktanın fiyatı, %
input int                   InpDate3=50;             // 3-üncü noktanın tarihi, %
```

```
input int                      InpPrice3=40;            // 3-üncü noktanın fiyatı, %
input ENUM_ELLIOT_WAVE_DEGREE InpDegree=ELLIOTT_MINOR; // Seviye
input bool                     InpDrawLines=true;       // Çizgilerin görüntülenmesi
input color                    InpColor=clrRed;         // Çizgilerin rengi
input ENUM_LINE_STYLE          InpStyle=STYLE_DASH;     // Çizgilerin stili
input int                      InpWidth=2;              // Çizgilerin kalınlığı
input bool                     InpBack=false;           // Arka-plan nesnesi
input bool                     InpSelection=true;       // Taşıma için vurgula
input bool                     InpHidden=true;          // Nesne listesinde gizle
input long                     InpZOrder=0;             // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Elliott Düzeltme Dalgasını verilen koordinatlarda oluştur        |
//+------------------------------------------------------------------+
bool ElliotWave3Create(const long                   chart_ID=0,              // çizel
                       const string                 name="ElliotWave3",      // dalga
                       const int                    sub_window=0,            // alt-p
                       datetime                     time1=0,                 // ilk n
                       double                       price1=0,                // ilk n
                       datetime                     time2=0,                 // ikinc
                       double                       price2=0,                // ikinc
                       datetime                     time3=0,                 // üçünc
                       double                       price3=0,                // üçünc
                       const ENUM_ELLIOT_WAVE_DEGREE degree=ELLIOTT_MINUETTE, // derec
                       const bool                   draw_lines=true,         // çizgi
                       const color                  clr=clrRed,              // nesne
                       const ENUM_LINE_STYLE        style=STYLE_SOLID,       // çizgi
                       const int                    width=1,                 // çizgi
                       const bool                   back=false,              // arka-
                       const bool                   selection=true,          // taşın
                       const bool                   hidden=true,             // nesne
                       const long                   z_order=0)               // fare
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeElliotWave3EmptyPoints(time1,price1,time2,price2,time3,price3);
//--- hata değerini sıfırla
   ResetLastError();
//--- Verilen koordinatlarda "Elliott Düzeltme Dalgasını" oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIOTWAVE3,sub_window,time1,price1,time2,price2
     {
      Print(__FUNCTION__,
            ": \"Elliott Düzeltme Dalgası\" oluşturulamadı! Hata kodu = ",GetLastError
      return(false);
     }
//--- dereceyi (dalga boyunu) ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_DEGREE,degree);
//--- çizgilerin görüntülenmesi modunu etkinleştir (true) veya devre dışı bırak (false
   ObjectSetInteger(chart_ID,name,OBJPROP_DRAWLINES,draw_lines);
//--- nesnenin rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
```

```
//--- çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- çizgilerin genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Elliott Düzeltme Dalgasının tutturma noktasını taşı              |
//+------------------------------------------------------------------+
bool ElliotWave3PointChange(const long   chart_ID=0,          // çizelge kimliği
                            const string name="ElliotWave3", // nesne ismi
                            const int    point_index=0,      // tutturma (çapa) noktas
                            datetime     time=0,             // tutturma noktası zaman
                            double       price=0)            // tutturma noktası fiyat
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Elliott Düzeltme Dalgasını sil                                   |
//+------------------------------------------------------------------+
bool ElliotWave3Delete(const long   chart_ID=0,          // çizelge kimliği
```

```
                        const string name="ElliotWave3") // nesne ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesneyi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Elliott Düzeltme Dalgası\" nesnesi silinemedi! Hata kodu = ",GetLastE
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Elliott Düzeltme Dalgasının tutturma noktalarının değerlerini    |
//| kontrol et ve boş olanlar için varsayılan değerleri ayarla       |
//+------------------------------------------------------------------+
void ChangeElliotWave3EmptyPoints(datetime &time1,double &price1,
                                  datetime &time2,double &price2,
                                  datetime &time3,double &price3)
  {
//--- son 10 çubuğun açılış zamanlarını almak için bir dizi
   datetime temp[];
   ArrayResize(temp,10);
//--- veriyi al
   CopyTime(Symbol(),Period(),TimeCurrent(),10,temp);
//--- çizelge üzerindeki bir noktanın değerini al
   double point=SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- ilk noktanın zamanı ayarlanmamışsa, son çubuktan 9 çubuk solda olacak
   if(!time1)
      time1=temp[0];
//--- ilk noktanın fiyatı ayarlanmamışsa mevcut Bid değerini alacak
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- ikinci noktanın zamanı ayarlanmamışsa, son çubuktan 5 çubuk solda olacak
   if(!time2)
      time2=temp[4];
//--- ikinci noktanın fiyatı ayarlanmamışsa, ilkinin 300 puan aşağısına konumla
   if(!price2)
      price2=price1-300*point;
//--- üçüncü noktanın zamanı ayarlanmamışsa, son çubuktan 1 çubuk solda olacak
   if(!time3)
      time3=temp[8];
//--- üçüncü noktanın fiyatı ayarlanmamışsa, ilkinin 200 puan aşağısına konumla
   if(!price3)
      price3=price1-200*point;
  }
//+------------------------------------------------------------------+
```

```
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   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("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- Elliott Düzeltme Dalgalarını çizmek için noktaları ayarla
   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;
//--- Elliott Düzeltme Dalgalarını oluştur
   if(!ElliotWave3Create(0,InpName,0,date[d1],price[p1],date[d2],price[p2],date[d3],pr
      InpDegree,InpDrawLines,InpColor,InpStyle,InpWidth,InpBack,InpSelection,InpHidder
     {
```

```
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, tutturma noktalarını taşı
//--- döngü sayacı
   int v_steps=accuracy/5;
//--- üçüncü tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p3<accuracy-1)
         p3+=1;
      //--- tutturma noktasını taşı
      if(!ElliotWave3PointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   v_steps=accuracy*4/5;
//--- ilk ve ikinci tutturma noktalarını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- şu değerleri kullan
      if(p1>1)
         p1-=1;
      if(p2<accuracy-1)
         p2+=1;
      //--- noktaları kaydır
      if(!ElliotWave3PointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!ElliotWave3PointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- nesneyi çizelgeden sil
```

```
   ElliotWave3Delete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_RECTANGLE

Dikdörtgen.

Not

Dikdörtgen için renk ile doldurma modu, OBJPROP_FILL özelliği ile seçilebilir.

Örnek

Aşağıdaki script, çizelge üzerinde bir dikdörtgen oluşturur ve onu taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, çizelge üzerinde bir dikdörtgen oluşturur."
#property description "Tutturma (çapa) noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string           InpName="Rectangle"; // Dikdörtgen adı
input int              InpDate1=40;         // 1-inci nokta tarihi, %
input int              InpPrice1=40;        // 1-inci nokta fiyatı, %
input int              InpDate2=60;         // 2-inci nokta tarihi, %
input int              InpPrice2=60;        // 2-inci nokta fiyatı, %
input color            InpColor=clrRed;     // Dikdörtgen rengi
input ENUM_LINE_STYLE  InpStyle=STYLE_DASH; // Dikdörtgen çizgi stili
input int              InpWidth=2;          // Dikdörtgen çizgi kalınlığı
```

```
input bool            InpFill=true;        // Dikdörtgenin renk ile doldurulması
input bool            InpBack=false;       // Arkaplan nesnesi
input bool            InpSelection=true;   // Taşımak için vurgula
input bool            InpHidden=true;      // Nesne listesinde gizle
input long            InpZOrder=0;         // Fare tıklaması önceliği
//+------------------------------------------------------------------+
//| Verilen koordinatlarda dikdörtgeni oluştur                       |
//+------------------------------------------------------------------+
bool RectangleCreate(const long            chart_ID=0,        // çizelge kimliği
                     const string          name="Rectangle",  // dikdörtgen ismi
                     const int             sub_window=0,      // alt pencere indisi
                     datetime              time1=0,           // ilk nokta zamanı
                     double                price1=0,          // ilk nokta fiyatı
                     datetime              time2=0,           // ikinci nokta zamanı
                     double                price2=0,          // ikinci nokta fiyatı
                     const color           clr=clrRed,        // dikdörtgen ismi
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // dikdörtgen çizgi stil
                     const int             width=1,           // dikdörtgen çizgi geni
                     const bool            fill=false,        // dikdörtgeni renk ile
                     const bool            back=false,        // arka-plan çizimi
                     const bool            selection=true,    // taşıma için vurgula
                     const bool            hidden=true,       // nesne listesinde giz
                     const long            z_order=0)         // fare tıklaması için
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeRectangleEmptyPoints(time1,price1,time2,price2);
//--- hata değerini sıfırla
   ResetLastError();
//--- verilen koordinatlarda bir dikdörtgen oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE,sub_window,time1,price1,time2,price2))
     {
      Print(__FUNCTION__,
            ": dikdörtgenin oluşturulması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- dikdörtgen rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- dikdörtgen çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- dikdörtgen çizgi genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- dikdörtgeni doldurma modunu etkinleştir (true) veya devre dışı bırak (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Dikdörtgenin tutturma noktasını taşı                             |
//+------------------------------------------------------------------+
bool RectanglePointChange(const long   chart_ID=0,       // çizelge kimliği
                          const string name="Rectangle", // dikdörtgen ismi
                          const int    point_index=0,    // tutturma noktası indisi
                          datetime     time=0,           // tutturma noktası zaman koo
                          double       price=0)          // tutturma noktası fiyat koo
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Dikdörtgeni sil                                                  |
//+------------------------------------------------------------------+
bool RectangleDelete(const long   chart_ID=0,       // çizelge kimliği
                     const string name="Rectangle") // dikdörtgen ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- dikdörtgeni sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": dikdörtgenin silinmesi başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
```

```
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Dikdörtgenin tutturma noktası değerlerini kontrol et ve          |
//| boş olanlar için varsayılan değerleri kullan                     |
//+------------------------------------------------------------------+
void ChangeRectangleEmptyPoints(datetime &time1,double &price1,
                                datetime &time2,double &price2)
  {
//--- ilk noktanın zamanı ayarlanmamışsa mevcut çubuk üzerinde olacak
   if(!time1)
      time1=TimeCurrent();
//--- ilk noktanın fiyatı ayarlanmamışsa mevcut Bid değerini alacak
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- ikinci noktanın zamanı ayarlanmamışsa, ilkinin 9 çubuk sonrasına konumlandırılı
   if(!time2)
     {
      //--- son 10 çubuğun açılış zamanını alacak bir dizi
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- ikinci noktayı birincisinden 9 çubuk ileri ayarla
      time2=temp[0];
     }
//--- ikinci noktanın fiyatı ayarlanmamışsa, ilkinin 300 puan aşağısına konumla
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate1<0 || InpDate1>100 || InpPrice1<0 || InpPrice1>100 ||
      InpDate2<0 || InpDate2>100 || InpPrice2<0 || InpPrice2>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
```

```
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- dikdörtgenin çizimi için noktaları tanımla
   int d1=InpDate1*(bars-1)/100;
   int d2=InpDate2*(bars-1)/100;
   int p1=InpPrice1*(accuracy-1)/100;
   int p2=InpPrice2*(accuracy-1)/100;
//--- bir dikdörtgen oluştur
   if(!RectangleCreate(0,InpName,0,date[d1],price[p1],date[d2],price[p2],InpColor,
      InpStyle,InpWidth,InpFill,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, dikdörtgenin tutturma noktalarını taşı
//--- döngü sayacı
   int h_steps=bars/2;
//--- tutturma noktalarını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- şu değerleri kullan
      if(d1<bars-1)
         d1+=1;
      if(d2>1)
         d2-=1;
      //--- noktaları kaydır
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
```

```
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   int v_steps=accuracy/2;
//--- tutturma noktalarını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- şu değerleri kullan
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- noktaları kaydır
      if(!RectanglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!RectanglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- dikdörtgeni çizelgeden sil
   RectangleDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_TRIANGLE

Üçgen.

Not

Üçgen için renk ile doldurma modu, OBJPROP_FILL özelliği kullanılarak ayarlanabilir.

Örnek

Aşağıdaki script, üçgeni çizelge üzerinde oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, üçgeni çizelge üzerinde oluşturur."
#property description "Tutturma (çapa) noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="Triangle";        // Üçgen ismi
input int             InpDate1=25;               // 1-inci noktanın tarihi, %
input int             InpPrice1=50;              // 1-inci noktanın fiyatı, %
input int             InpDate2=70;               // 2-inci noktanın tarihi, %
input int             InpPrice2=70;              // 2-inci noktanın fiyatı, %
input int             InpDate3=65;               // 3-üncü noktanın tarihi, %
input int             InpPrice3=20;              // 3-üncü noktanın fiyatı, %
input color           InpColor=clrRed;           // Üçgen rengi
```

```
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Üçgen çizgilerinin stili
input int             InpWidth=2;                // Üçgen çizgilerinin kalınlığı
input bool            InpFill=false;             // Üçgeni renk ile doldur
input bool            InpBack=false;             // Arkaplan nesnesi
input bool            InpSelection=true;         // Taşımak için vurgula
input bool            InpHidden=true;            // Nesne listesinde gizle
input long            InpZOrder=0;               // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Verilen koordinatlarda üçgeni oluştur                            |
//+------------------------------------------------------------------+
bool TriangleCreate(const long            chart_ID=0,        // çizelge kimliği
                    const string          name="Triangle",   // üçgen ismi
                    const int             sub_window=0,      // alt-pencere indisi
                    datetime              time1=0,           // ilk nokta zamanı
                    double                price1=0,          // ilk nokta fiyatı
                    datetime              time2=0,           // ikinci nokta zamanı
                    double                price2=0,          // ikinci nokta fiyatı
                    datetime              time3=0,           // üçüncü nokta zamanı
                    double                price3=0,          // üçüncü nokta fiyatı
                    const color           clr=clrRed,        // üçgen rengi
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // üçgen çizgilerinin sti
                    const int             width=1,           // üçgen çizgilerinin ger
                    const bool            fill=false,        // üçgeni renk ile doldu
                    const bool            back=false,        // arka planda
                    const bool            selection=true,    // taşıma için vurgula
                    const bool            hidden=true,       // nesneyi listede gizle
                    const long            z_order=0)         // fare tıklaması için ön
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeTriangleEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- hata değerini sıfırla
   ResetLastError();
//--- üçgeni, verilen koordinatlarda oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_TRIANGLE,sub_window,time1,price1,time2,price2,ti
     {
      Print(__FUNCTION__,
            ": üçgenin oluşturulması başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- üçgenin rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- üçgen çizgilerinin stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- üçgen çizgilerinin kalınlığını ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- üçgeni doldurma modunu etkinleştir (true) veya devre dışı bırak (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
```

```
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Üçgenin tutturma noktasını taşı                                  |
//+------------------------------------------------------------------+
bool TrianglePointChange(const long   chart_ID=0,      // çizelge kimliği
                         const string name="Triangle", // üçgenin ismi
                         const int    point_index=0,   // tutturma noktası indisi
                         datetime     time=0,          // çapa noktası zaman koordinat
                         double       price=0)         // çapa noktası fiyat koordinat
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğ
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Üçgeni sil                                                       |
//+------------------------------------------------------------------+
bool TriangleDelete(const long   chart_ID=0,      // çizelge kimliği
                    const string name="Triangle") // üçgenin ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- üçgeni sil
   if(!ObjectDelete(chart_ID,name))
```

```
     {
      Print(__FUNCTION__,
            ": üçgenin silinmesi başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Üçgenin tutturma noktası değerlerini kontrol et ve               |
//| boş olanlar için varsayılan değerleri kullan                     |
//+------------------------------------------------------------------+
void ChangeTriangleEmptyPoints(datetime &time1,double &price1,
                               datetime &time2,double &price2,
                               datetime &time3,double &price3)
  {
//--- ilk noktanın zamanı ayarlanmamışsa mevcut çubuk üzerinde olacak
   if(!time1)
      time1=TimeCurrent();
//--- ilk noktanın fiyatı ayarlanmamışsa mevcut Bid değerini alacak
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- ikinci noktanın zamanı ayarlanmamışsa, ilkinin 9 çubuk sonrasına konumlandırılı
   if(!time2)
     {
      //--- son 10 çubuğun açılış zamanını alacak bir dizi
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- ikinci noktayı birincisinden 9 çubuk ileri ayarla
      time2=temp[0];
     }
//--- ikinci noktanın fiyatı ayarlanmamışsa, ilkinin 300 puan aşağısına konumla
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- üçüncü noktanın zamanı ayarlanmamışsa, ikincinin tarihine karşılık gelir
   if(!time3)
      time3=time2;
//--- üçüncü noktanın fiyatı ayarlanmamışsa, ilk noktanınkine eşittir
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   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("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- üçgeni çizmek için noktaları tanımla
   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;
//--- bir üçgen oluştur
   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;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, üçgenin taşıma noktalarını taşı
//--- döngü sayacı
```

```
   int v_steps=accuracy*3/10;
//--- ilk tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p1>1)
         p1-=1;
      //--- tutturma noktasını taşı
      if(!TrianglePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   int h_steps=bars*9/20-1;
//--- ikinci tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d2>1)
         d2-=1;
      //--- tutturma noktasını taşı
      if(!TrianglePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   v_steps=accuracy/4;
//--- üçüncü tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p3<accuracy-1)
         p3+=1;
      //--- tutturma noktasını taşı
      if(!TrianglePointChange(0,InpName,2,date[d3],price[p3]))
```

```
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- elipsi çizelgeden sil
   TriangleDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_ELLIPSE

Elips.

Not

Elips için renkle doldurma modu, OBJPROP_FILL özelliği ile ayarlanabilir.

Örnek

Aşağıdaki script, çizelge üzerinde elips nesnesini oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, çizelge üzerinde elips şekli oluşturur."
#property description "tutturma koordinatları, pencere boyutunun"
#property description "yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="Ellipse";         // Elipsin ismi
input int             InpDate1=30;               // 1-inci noktanın tarihi, %
input int             InpPrice1=20;              // 1-inci noktanın fiyatı, %
input int             InpDate2=70;               // 2-inci noktanın tarihi, %
input int             InpPrice2=80;              // 2-inci noktanın fiyatı, %
input int             InpDate3=50;               // 3-üncü noktanın tarihi, %
input int             InpPrice3=60;              // 3-üncü noktanın fiyatı, %
input color           InpColor=clrRed;           // Elips rengi
input ENUM_LINE_STYLE InpStyle=STYLE_DASHDOTDOT; // Elips çizgi stili
```

```
input int             InpWidth=2;                // Elips çizgilerinin kalınlığı
input bool            InpFill=false;             // Elipsi renkle doldurma
input bool            InpBack=false;             // Arka-plan elipsi
input bool            InpSelection=true;         // Taşımak için vurgula
input bool            InpHidden=true;            // Nesne listesinde gizle
input long            InpZOrder=0;               // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Verilen koordinatlarda bir elips oluştur                         |
//+------------------------------------------------------------------+
bool EllipseCreate(const long            chart_ID=0,        // çizelge kimliği
                   const string          name="Ellipse",    // elips ismi
                   const int             sub_window=0,      // alt-pencere indisi
                   datetime              time1=0,           // ilk noktanın zamanı
                   double                price1=0,          // ilk noktanın fiyatı
                   datetime              time2=0,           // ikinci nokta zamanı
                   double                price2=0,          // ikinci nokta fiyatı
                   datetime              time3=0,           // üçüncü nokta zamanı
                   double                price3=0,          // üçüncü nokta fiyatı
                   const color           clr=clrRed,        // elips rengi
                   const ENUM_LINE_STYLE style=STYLE_SOLID, // elips çizgilerinin stil
                   const int             width=1,           // elips çizgi kalınlığı
                   const bool            fill=false,        // elipsi renk ile doldurr
                   const bool            back=false,        // arka-planda
                   const bool            selection=true,    // taşıma için vurgula
                   const bool            hidden=true,       // nesne listesinde gizle
                   const long            z_order=0)         // fare tıklaması için önc
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeEllipseEmptyPoints(time1,price1,time2,price2,time3,price3);
//--- hata değerini sıfırla
   ResetLastError();
//--- verilen koordinatlarda bir elips oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_ELLIPSE,sub_window,time1,price1,time2,price2,tir
     {
      Print(__FUNCTION__,
            ": elipsin oluşturulması başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- bir elips rengi ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- elips çizgilerinin stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- elips çizgilerinin kalınlığını ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- elips doldurma modunu etkinleştir (true) veya devre dışı bırak (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_FILL,fill);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- taşıma için vurgulama modunu etkinleştir (true) veya devre dışı bırak (false)
```

```
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Elipsin tutturma noktasını taşı                                  |
//+------------------------------------------------------------------+
bool EllipsePointChange(const long   chart_ID=0,     // çizelge kimliği
                        const string name="Ellipse", // elips ismi
                        const int    point_index=0,  // tutturma noktası indisi
                        datetime     time=0,         // tutturma noktası zaman koordi
                        double       price=0)        // tutturma noktası fiyat koordi
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğ
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,point_index,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Elipsi sil                                                       |
//+------------------------------------------------------------------+
bool EllipseDelete(const long   chart_ID=0,     // çizelge kimliği
                   const string name="Ellipse") // elips ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- elipsi sil
   if(!ObjectDelete(chart_ID,name))
     {
```

```
      Print(__FUNCTION__,
            ": elipsin silinmesi başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Elipsin tutturma (çapa) noktalarının değerlerini kontrol et      |
//| boş olanlar için varsayılan değerleri ayarla                     |
//+------------------------------------------------------------------+
void ChangeEllipseEmptyPoints(datetime &time1,double &price1,
                              datetime &time2,double &price2,
                              datetime &time3,double &price3)
  {
//--- ilk noktanın zamanı ayarlanmamışsa mevcut çubuk üzerinde olacak
   if(!time1)
      time1=TimeCurrent();
//--- ilk noktanın fiyatı ayarlanmamışsa mevcut Bid değerini alacak
   if(!price1)
      price1=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- ikinci noktanın zamanı ayarlanmamışsa, ilkinin 9 çubuk sonrasına konumlandırılır
   if(!time2)
     {
      //--- son 10 çubuğun açılış zamanını alacak bir dizi
      datetime temp[10];
      CopyTime(Symbol(),Period(),time1,10,temp);
      //--- ikinci noktayı birincisinden 9 çubuk ileri ayarla
      time2=temp[0];
     }
//--- ikinci noktanın fiyatı ayarlanmamışsa, ilkinin 300 puan aşağısına konumla
   if(!price2)
      price2=price1-300*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
//--- üçüncü noktanın zamanı ayarlanmamışsa, ikincinin tarihine karşılık gelir
   if(!time3)
      time3=time2;
//--- üçüncü noktanın fiyatı ayarlanmamışsa, ilk noktanınkine eşittir
   if(!price3)
      price3=price1;
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   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("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih değerlerini depolayacak bir dizi
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- elipsin çizileceği noktaları ayarla
   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;
//--- bir elips oluştur
   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;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, elipsin taşıma noktalarını taşı
//--- döngü sayacı
   int v_steps=accuracy/5;
```

```
//--- ilk ve ikinci tutturma noktalarını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- şu değerleri kullan
      if(p1<accuracy-1)
         p1+=1;
      if(p2>1)
         p2-=1;
      //--- noktaları kaydır
      if(!EllipsePointChange(0,InpName,0,date[d1],price[p1]))
         return;
      if(!EllipsePointChange(0,InpName,1,date[d2],price[p2]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   int h_steps=bars/5;
//--- üçüncü tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d3>1)
         d3-=1;
      //--- tutturma noktasını taşı
      if(!EllipsePointChange(0,InpName,2,date[d3],price[p3]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- elipsi çizelgeden sil
   EllipseDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_ARROW_THUMB_UP

Başparmak Yukarı işareti.

Not

İşarete göre tutturma konumu ENUM_ARROW_ANCHOR sayımından seçilebilir.

MetaEditor'de bir kod yazarken büyük işaretler (5'ten büyük), sadece uygun OBJPROP_WIDTH özellik değerini ayarlayarak oluşturulabilir.

Örnek

Aşağıdaki script, Başparmak Yukarı işaretini oluşturur ve onu çizelge üzerinde taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Başparmak Yukarı\" işaretini çizer."
#property description "Tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string             InpName="ThumbUp";     // İşaret ismi
input int                InpDate=75;            // % şeklinde tutturma noktası tarihi
input int                InpPrice=25;           // % şeklinde tutturma noktası fiyatı
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP;  // Tutturma noktası tipi
input color              InpColor=clrRed;       // İşaret rengi
input ENUM_LINE_STYLE    InpStyle=STYLE_DOT;    // Kenar çizgisi stili
```

```
input int                InpWidth=5;           // İşaret boyutu
input bool               InpBack=false;        // Arkaplan işareti
input bool               InpSelection=true;    // Taşıma için vurgula
input bool               InpHidden=true;       // Nesne listesinde gizle
input long               InpZOrder=0;          // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Başparmak Yukarı işaretini oluştur                               |
//+------------------------------------------------------------------+
bool ArrowThumbUpCreate(const long              chart_ID=0,           // çizelge tanır
                        const string            name="ThumbUp",       // işaret ismi
                        const int               sub_window=0,         // alt pencere i
                        datetime                time=0,               // tutturma nokt
                        double                  price=0,              // tutturma nokt
                        const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // çapa (tutturm
                        const color             clr=clrRed,           // işaret rengi
                        const ENUM_LINE_STYLE   style=STYLE_SOLID,    // kenar çizgisi
                        const int               width=3,              // işaret boyutu
                        const bool              back=false,           // arka-planda ç
                        const bool              selection=true,       // taşıma için v
                        const bool              hidden=true,          // nesne listesi
                        const long              z_order=0)            // fare tıklamas
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeArrowEmptyPoint(time,price);
//--- hata değerini sıfırla
   ResetLastError();
//--- işareti oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"Başparmak Yukarı\" işaretinin oluşturulması başarısız! Hata kodu = ",
      return(false);
     }
//--- tutturma noktası tipini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- bir işaret rengi ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- kenar çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- işaret boyutunu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- işareti fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
```

```
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktasını taşı                                          |
//+------------------------------------------------------------------+
bool ArrowThumbUpMove(const long   chart_ID=0,      // çizelge kimliği
                      const string name="ThumbUp", // nesne ismi
                      datetime     time=0,         // tutturma noktası zaman koordinat
                      double       price=0)        // tutturma noktası fiyat koordinat
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Başparmak Yukarı işaretinin tutturma noktası ayarla              |
//+------------------------------------------------------------------+
bool ArrowThumbUpAnchorChange(const long              chart_ID=0,        // çizelge ki
                              const string            name="ThumbUp",    // nesne ismi
                              const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // çapa (tutt
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktası tipini değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": tutturma noktası tipinin değiştirilmesi başarısız oldu! Hata kodu = ",G
      return(false);
     }
//--- başarılı çalıştırma
```

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Başparmak Yukarı işaretini sil                                   |
//+------------------------------------------------------------------+
bool ArrowThumbUpDelete(const long   chart_ID=0,     // çizelge kimliği
                        const string name="ThumbUp") // işaret ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- işareti sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Başparmak Yukarı\" işaretinin silinmesi başarısız! Hata kodu = ",GetL
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktası değerlerini kontrol et ve                       |
//| boş olanlar için varsayılan değerleri ayarla                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- noktanın zamanı ayarlanmamışsa, mevcut çubuk üzerinde olacak
   if(!time)
      time=TimeCurrent();
//--- noktanın fiyatı ayarlanmamışsa, Bid değerini alacak
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
```

```
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- işaretin çizileceği noktaları tanımla
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- Başparmak Yukarı işaretini çizelge üzerinde oluştur
   if(!ArrowThumbUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, tutturma noktasını taşı ve konumunu işarete göre değiştir
//--- döngü sayacı
   int h_steps=bars/4;
//--- tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d>1)
         d-=1;
      //--- tutturma noktasını taşı
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
```

```
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   int v_steps=accuracy/4;
//--- tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p<accuracy-1)
         p+=1;
      //--- tutturma noktasını taşı
      if(!ArrowThumbUpMove(0,InpName,date[d],price[p]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- tutturma noktasının konumunu işarete göre değiştir
   ArrowThumbUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- çizelgeyi yeniden çiz
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- işareti çizelgeden sil
   ArrowThumbUpDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_ARROW_THUMB_DOWN

Başparmak Aşağı (kötü) işareti.

Not

İşarete göre tutturma konumu ENUM_ARROW_ANCHOR sayımından seçilebilir.

MetaEditor'de bir kod yazarken büyük işaretler (5'ten büyük), sadece uygun OBJPROP_WIDTH özellik değerini ayarlayarak oluşturulabilir.

Örnek

Aşağıdaki script, Başparmak Aşağı işaretini oluşturur ve onu çizelge üzerinde taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script \"Başparmak Aşağı\" işaretini çizer."
#property description "Tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string             InpName="ThumbDown";      // İşaret ismi
input int                InpDate=25;               // % şeklinde tutturma noktası tarihi
input int                InpPrice=75;              // % şeklinde tutturma noktası fiyatı
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_BOTTOM;  // Tutturma tipi
input color              InpColor=clrRed;          // İşaret rengi
```

```
input ENUM_LINE_STYLE   InpStyle=STYLE_DOT;      // Kenar çizgisi stili
input int               InpWidth=5;              // İşaret boyutu
input bool              InpBack=false;           // Arkaplan işareti
input bool              InpSelection=true;       // Taşıma için vurgula
input bool              InpHidden=true;          // Nesne listesinde gizleme
input long              InpZOrder=0;             // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Başparmak Aşağı işaretini oluştur                                |
//+------------------------------------------------------------------+
bool ArrowThumbDownCreate(const long              chart_ID=0,           // çizelge kim
                          const string            name="ThumbDown",     // işaret ismi
                          const int               sub_window=0,         // alt-pencere
                          datetime                time=0,               // tutturma no
                          double                  price=0,              // tutturma no
                          const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // tutturma ti
                          const color             clr=clrRed,           // işaret renç
                          const ENUM_LINE_STYLE   style=STYLE_SOLID,    // kenar çizgi
                          const int               width=3,              // işaret büyü
                          const bool              back=false,           // arka-planda
                          const bool              selection=true,       // taşıma için
                          const bool              hidden=true,          // nesne liste
                          const long              z_order=0)            // fare tıklan
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeArrowEmptyPoint(time,price);
//--- hata değerini sıfırla
   ResetLastError();
//--- işareti oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_THUMB_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"Başparmak Aşağı\" işaretinin oluşturulması başarısız oldu! Hata kodu
      return(false);
     }
//--- tutturma noktası tipini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- bir işaret rengi ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- kenar çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- işaret boyutunu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- işareti fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktasını taşı                                          |
//+------------------------------------------------------------------+
bool ArrowThumbDownMove(const long   chart_ID=0,        // çizelge kimliği
                        const string name="ThumbDown", // nesne ismi
                        datetime     time=0,           // tutturma noktasının zaman ko
                        double       price=0)          // tutturma noktasının fiyat ko
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Başparmak Aşağı işaretinin çapa tipi değiştir                    |
//+------------------------------------------------------------------+
bool ArrowThumbDownAnchorChange(const long              chart_ID=0,        // çizelge
                                const string            name="ThumbDown",  // nesne is
                                const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // çapa tip
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktası tipini değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": tutturma noktası tipinin değiştirilmesi başarısız oldu! Hata kodu = ",G
      return(false);
     }
```

```
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Başparmak Aşağı işaretini sil                                    |
//+------------------------------------------------------------------+
bool ArrowThumbDownDelete(const long   chart_ID=0,       // çizelge kimliği
                          const string name="ThumbDown") // işaret ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- işareti sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Başparmak Aşağı\" işaretinin silinmesi başarısız oldu! Hata kodu = ",
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktası değerlerini kontrol et ve                       |
//| boş olanlar için varsayılan değerleri ayarla                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- noktanın zamanı ayarlanmamışsa, mevcut çubuk üzerinde olacak
   if(!time)
      time=TimeCurrent();
//--- noktanın fiyatı ayarlanmamışsa, Bid değerini alacak
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
```

```
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- işaretin çizileceği noktaları tanımla
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- Başparmak Aşağı işaretini çizelge üzerinde oluştur
   if(!ArrowThumbDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, tutturma noktasını taşı ve konumunu işarete göre değiştir
//--- döngü sayacı
   int h_steps=bars/4;
//--- tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d<bars-1)
         d+=1;
      //--- tutturma noktasını taşı
      if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
```

```
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- döngü sayacı
   int v_steps=accuracy/4;
//--- tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p>1)
         p-=1;
      //--- tutturma noktasını taşı
      if(!ArrowThumbDownMove(0,InpName,date[d],price[p]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- tutturma noktasının konumunu işarete göre değiştir
   ArrowThumbDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- çizelgeyi yeniden çiz
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- işareti çizelgeden sil
   ArrowThumbDownDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_ARROW_UP

Yukarı Ok işareti.

Not

İşarete göre tutturma konumu ENUM_ARROW_ANCHOR sayımından seçilebilir.

MetaEditor'de bir kod yazarken büyük işaretler (5'ten büyük), sadece uygun OBJPROP_WIDTH özellik değerini ayarlayarak oluşturulabilir.

Örnek

Aşağıdaki script, Yukarı Ok işaretini oluşturur ve onu çizelge üzerinde taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Bertik, \"Yukarı Ok\" işaretini çizer."
#property description "tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string             InpName="ArrowUp";     // İşaret ismi
input int                InpDate=25;            // % şeklinde tutturma noktası tarihi
input int                InpPrice=25;           // % şeklinde tutturma noktası fiyatı
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP; // Tutturma noktası tipi
input color              InpColor=clrRed;       // İşaret rengi
input ENUM_LINE_STYLE    InpStyle=STYLE_DOT;    // Kenar çizgisi stili
```

```
input int               InpWidth=5;           // İşaret boyutu
input bool              InpBack=false;        // Arkaplan işareti
input bool              InpSelection=false;   // taşıma için vurgula
input bool              InpHidden=true;       // Nesne listesinde gizle
input long              InpZOrder=0;          // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Yukarı Ok işaretini oluştur                                      |
//+------------------------------------------------------------------+
bool ArrowUpCreate(const long              chart_ID=0,           // çizelge kimliği
                   const string            name="ArrowUp",       // işaret ismi
                   const int               sub_window=0,         // alt-pencere indisi
                   datetime                time=0,               // tutturma noktası z
                   double                  price=0,              // tutturma noktası f
                   const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // çapa tipi
                   const color             clr=clrRed,           // işaret rengi
                   const ENUM_LINE_STYLE   style=STYLE_SOLID,    // kenar çizgisi kali
                   const int               width=3,              // işaret boyutu
                   const bool              back=false,           // arka-plana çiz
                   const bool              selection=true,       // taşıma için vurgul
                   const bool              hidden=true,          // nesne listesinde g
                   const long              z_order=0)            // fare tıklaması içi
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeArrowEmptyPoint(time,price);
//--- hata değerini sıfırla
   ResetLastError();
//--- işareti oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_UP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"Yukarı Ok\" işaretini oluşturma başarısız! Hata kodu = ",GetLastErro
      return(false);
     }
//--- tutturma noktası tipini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- bir işaret rengi ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- kenar çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- işaret boyutunu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- işareti fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
```

```
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktasını taşı                                          |
//+------------------------------------------------------------------+
bool ArrowUpMove(const long   chart_ID=0,     // çizelge tanımlayıcısı
                 const string name="ArrowUp", // nesne ismi
                 datetime     time=0,         // tutturma (çapa) noktası zaman koordi
                 double       price=0)        // tutturma noktası fiyat koordinatı
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğ
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Aşağı Ok işaretinin tutturma tipini değiştir                     |
//+------------------------------------------------------------------+
bool ArrowUpAnchorChange(const long              chart_ID=0,        // çizelge tanımla
                         const string            name="ArrowUp",    // nesne ismi
                         const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // çapa tipi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktası konumunu değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": tutturma noktası tipinin değiştirilmesi başarısız oldu! Hata kodu = ",G
      return(false);
     }
//--- başarılı çalıştırma
```

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Yukarı Ok işaretini sil                                          |
//+------------------------------------------------------------------+
bool ArrowUpDelete(const long   chart_ID=0,     // çizelge kimliği
                   const string name="ArrowUp") // işaret ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- işareti sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Yukarı Ok\" işaretinin silinmesi başarısız! Hata kodu = ",GetLastErro
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktası değerlerini kontrol et ve                       |
//| boş olanlar için varsayılan değerleri ayarla                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- noktanın zamanı ayarlanmamışsa, mevcut çubuk üzerinde olacak
   if(!time)
      time=TimeCurrent();
//--- noktanın fiyatı ayarlanmamışsa, Bid değerini alacak
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
```

```
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- işaretin çizileceği noktaları tanımla
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- Yukarı Ok işaretini oluştur
   if(!ArrowUpCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, tutturma noktasını taşı ve konumunu işarete göre değiştir
//--- döngü sayacı
   int v_steps=accuracy/2;
//--- tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p<accuracy-1)
         p+=1;
      //--- tutturma noktasını taşı
      if(!ArrowUpMove(0,InpName,date[d],price[p]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
```

```
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- tutturma noktasının konumunu işarete göre değiştir
   ArrowUpAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- çizelgeyi yeniden çiz
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- işareti çizelgeden sil
   ArrowUpDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_ARROW_DOWN

Yukarı Ok işareti.

Not

İşarete göre tutturma konumu ENUM_ARROW_ANCHOR sayımından seçilebilir.

MetaEditor'de bir kod yazarken büyük işaretler (5'ten büyük), sadece uygun OBJPROP_WIDTH özellik değerini ayarlayarak oluşturulabilir.

Örnek

Aşağıdaki script, Aşağı Ok işaretini oluşturur ve onu çizelge üzerinde taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script \"Aşağı Ok\" işaretini çizer."
#property description "tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string             InpName="ArrowDown";      // İşaret ismi
input int                InpDate=75;               // % şeklinde tutturma noktası tarihi
input int                InpPrice=75;              // % şeklinde tutturma noktası fiyatı
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_BOTTOM;  // Tutturma noktası tipi
input color              InpColor=clrRed;          // İşaret rengi
input ENUM_LINE_STYLE    InpStyle=STYLE_DOT;       // Kenar çizgisi stili
input int                InpWidth=5;               // İşaret boyutu
```

```
input bool               InpBack=false;           // Arkaplan işareti
input bool               InpSelection=false;      // Taşıma için vurgula
input bool               InpHidden=true;          // Nesne listesinde gizleme
input long               InpZOrder=0;             // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Aşağı Ok işaretini oluştur                                       |
//+------------------------------------------------------------------+
bool ArrowDownCreate(const long              chart_ID=0,           // çizelge tanımla
                     const string            name="ArrowDown",     // işaret ismi
                     const int               sub_window=0,         // alt pencere ind
                     datetime                time=0,               // tutturma noktas
                     double                  price=0,              // tutturma noktas
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // tutturma noktas
                     const color             clr=clrRed,           // işaret rengi
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,    // kenar çizgisi s
                     const int               width=3,              // işaret boyutu
                     const bool              back=false,           // arka-plana al
                     const bool              selection=true,       // taşıma için vur
                     const bool              hidden=true,          // nesne listesind
                     const long              z_order=0)            // fare tıklaması
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeArrowEmptyPoint(time,price);
//--- hata değerini sıfırla
   ResetLastError();
//--- işareti oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_DOWN,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"Aşağı Ok\" işaretinin oluşturulması başarısız! Hata kodu = ",GetLastE
      return(false);
     }
//--- tutturma noktası tipi
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- bir işaret rengi ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- kenar çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- işaret boyutunu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- işareti fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktasını taşı                                          |
//+------------------------------------------------------------------+
bool ArrowDownMove(const long   chart_ID=0,        // çizelge kimliği
                   const string name="ArrowDown", // nesne ismi
                   datetime     time=0,           // tutturma noktasının zaman koordi
                   double       price=0)          // tutturma noktası fiyat koordinat
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğ
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Aşağı Ok işaretinin tutturma tipini değiştir                     |
//+------------------------------------------------------------------+
bool ArrowDownAnchorChange(const long              chart_ID=0,         // çizelgenin k
                           const string            name="ArrowDown",   // nesne ismi
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // tutturma tip
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktası konumunu değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": tutturma noktası tipinin değiştirilmesi başarısız oldu! Hata kodu = ",G
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
```

```
  }
//+------------------------------------------------------------------+
//| Aşağı Ok işaretini sil                                           |
//+------------------------------------------------------------------+
bool ArrowDownDelete(const long   chart_ID=0,       // çizelge kimliği
                     const string name="ArrowDown") // işaret ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- işareti sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Aşağı Ok\" işaretinin silinmesi başarısız! Hata kodu = ",GetLastErro
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktası değerlerini kontrol et ve                       |
//| boş olanlar için varsayılan değerleri ayarla                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- noktanın zamanı ayarlanmamışsa, mevcut çubuk üzerinde olacak
   if(!time)
      time=TimeCurrent();
//--- noktanın fiyatı ayarlanmamışsa, Bid değerini alacak
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
```

```
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- işaretin çizileceği noktaları tanımla
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- çizelge üzerinde Aşağı Ok işaretini oluştur
   if(!ArrowDownCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, tutturma noktasını taşı ve konumunu işarete göre değiştir
//--- döngü sayacı
   int v_steps=accuracy/2;
//--- tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p>1)
         p-=1;
      //--- tutturma noktasını taşı
      if(!ArrowDownMove(0,InpName,date[d],price[p]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
```

```
      }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- tutturma noktasının konumunu işarete göre değiştir
   ArrowDownAnchorChange(0,InpName,ANCHOR_TOP);
//--- çizelgeyi yeniden çiz
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- işareti çizelgeden sil
   ArrowDownDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_ARROW_STOP

Dur işareti.

Not

İşarete göre tutturma konumu ENUM_ARROW_ANCHOR sayımından seçilebilir.

MetaEditor'de bir kod yazarken büyük işaretler (5'ten büyük), sadece uygun OBJPROP_WIDTH özellik değerini ayarlayarak oluşturulabilir.

Örnek

Aşağıdaki script, Dur işaretini oluşturur ve çizelge üzerinde taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script \"Dur\" işaretini çizer."
#property description "tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string             InpName="ArrowStop";      // İşaret ismi
input int                InpDate=10;               // % şeklinde tutturma noktası tarihi
input int                InpPrice=50;              // % şeklinde tutturma noktası fiyatı
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_BOTTOM;  // Tutturma tipi
input color              InpColor=clrRed;          // İşaret rengi
input ENUM_LINE_STYLE    InpStyle=STYLE_DOT;       // Kenar çizgisi stili
```

```
input int                InpWidth=5;              // İşaret boyutu
input bool               InpBack=false;           // Arkaplan işareti
input bool               InpSelection=false;      // Taşıma için vurgula
input bool               InpHidden=true;          // Nesne listesinde gizleme
input long               InpZOrder=0;             // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Dur işaretini oluştur                                            |
//+------------------------------------------------------------------+
bool ArrowStopCreate(const long              chart_ID=0,           // çizelge kimliği
                     const string            name="ArrowStop",     // işaret ismi
                     const int               sub_window=0,         // alt pencere indi
                     datetime                time=0,               // tutturma noktasi
                     double                  price=0,              // tutturma noktasi
                     const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // tutturma noktasi
                     const color             clr=clrRed,           // işaret rengi
                     const ENUM_LINE_STYLE   style=STYLE_SOLID,    // kenar çizgisi st
                     const int               width=3,              // işaret boyutu
                     const bool              back=false,           // arka-plana al
                     const bool              selection=true,       // taşıma için vurç
                     const bool              hidden=true,          // nesne listesinde
                     const long              z_order=0)            // fare tıklaması i
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeArrowEmptyPoint(time,price);
//--- hata değerini sıfırla
   ResetLastError();
//--- işareti oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_STOP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"Dur işaretinin\" oluşturulması başarısız oldu! Hata kodu = ",GetLastE
      return(false);
     }
//--- tutturma noktası tipini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- bir işaret rengi ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- kenar çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- işaret boyutunu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- işareti fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
```

```
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktasını taşı                                          |
//+------------------------------------------------------------------+
bool ArrowStopMove(const long   chart_ID=0,       // çizelge tanımlayıcısı
                   const string name="ArrowStop", // nesne ismi
                   datetime     time=0,           // tutturma noktasının zaman koordi
                   double       price=0)          // tutturma noktası fiyat koordinat
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğ
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Dur işaretinin tutturma tipini değiştir                          |
//+------------------------------------------------------------------+
bool ArrowStopAnchorChange(const long              chart_ID=0,        // çizelge tanı
                           const string            name="ArrowStop",  // nesne ismi
                           const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // tutturma nok
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktası tipini değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": tutturma noktası tipinin değiştirilmesi başarısız oldu! Hata kodu = ",G
      return(false);
     }
//--- başarılı çalıştırma
```

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Dur işaretini sil                                                |
//+------------------------------------------------------------------+
bool ArrowStopDelete(const long   chart_ID=0,       // çizelge kimliği
                     const string name="ArrowStop") // etiket ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- işareti sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Dur\" işaretinin silinmesi başarısız oldu! Hata kodu = ",GetLastError
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktası değerlerini kontrol et ve                       |
//| boş olanlar için varsayılan değerleri ayarla                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- noktanın zamanı ayarlanmamışsa, mevcut çubuk üzerinde olacak
   if(!time)
      time=TimeCurrent();
//--- noktanın fiyatı ayarlanmamışsa, Bid değerini alacak
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
```

```
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- işaretin çizileceği noktaları tanımla
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- Dur işaretini çizelge üzerinde oluştur
   if(!ArrowStopCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, tutturma noktasını taşı ve konumunu işarete göre değiştir
//--- döngü sayacı
   int h_steps=bars*2/5;
//--- tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d<bars-1)
         d+=1;
      //--- tutturma noktasını taşı
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
```

```
      ChartRedraw();
      // 0.025 saniyelik gecikme
      Sleep(25);
     }
//--- tutturma noktasının konumunu işarete göre değiştir
   ArrowStopAnchorChange(0,InpName,ANCHOR_TOP);
//--- çizelgeyi yeniden çiz
   ChartRedraw();
//--- döngü sayacı
   h_steps=bars*2/5;
//--- tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d<bars-1)
         d+=1;
      //--- tutturma noktasını taşı
      if(!ArrowStopMove(0,InpName,date[d],price[p]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.025 saniyelik gecikme
      Sleep(25);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- işareti çizelgeden sil
   ArrowStopDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_ARROW_CHECK

Kontrol işareti.

Not

İşarete göre tutturma konumu ENUM_ARROW_ANCHOR sayımından seçilebilir.

MetaEditor'de bir kod yazarken büyük işaretler (5'ten büyük), sadece uygun OBJPROP_WIDTH özellik değerini ayarlayarak oluşturulabilir.

Örnek

Aşağıdaki script, Kontrol işaretini oluşturur ve çizelge üzerinde taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Bu script \"Kontrol\" işaretini çizer."
#property description "tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string             InpName="ArrowCheck"; // İşaret ismi
input int                InpDate=10;           // % şeklinde tutturma koordinatı tarihi
input int                InpPrice=50;          // % şeklinde tutturma koordinatı fiyatı
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP; // Tutturma noktası tipi
input color              InpColor=clrRed;      // İşaret rengi
input ENUM_LINE_STYLE    InpStyle=STYLE_DOT;   // Kenar çizgisi stili
```

```
input int                InpWidth=5;           // İşaret boyutu
input bool               InpBack=false;        // Arkaplan işareti
input bool               InpSelection=false;   // taşıma için vurgula
input bool               InpHidden=true;       // Nesne listesinde gizle
input long               InpZOrder=0;          // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Kontrol işareti oluştur                                          |
//+------------------------------------------------------------------+
bool ArrowCheckCreate(const long              chart_ID=0,           // çizelge tanımla
                      const string            name="ArrowCheck",    // işaret ismi
                      const int               sub_window=0,         // alt-pencere ind
                      datetime                time=0,               // tutturma noktas
                      double                  price=0,              // tutturma noktas
                      const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // tutturma noktas
                      const color             clr=clrRed,           // işaret rengi
                      const ENUM_LINE_STYLE   style=STYLE_SOLID,    // kenar çizgisi s
                      const int               width=3,              // işaret boyutu
                      const bool              back=false,           // arka planda çiz
                      const bool              selection=true,       // taşıma için vur
                      const bool              hidden=true,          // nesne listesind
                      const long              z_order=0)            // fare tıklaması
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeArrowEmptyPoint(time,price);
//--- hata değerini sıfırla
   ResetLastError();
//--- işareti oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_CHECK,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"Kontrol\" işaretinin oluşturulması başarısız oldu! Hata kodu = ",GetL
      return(false);
     }
//--- tutturma noktası tipini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- bir işaret rengi ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- kenar çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- işaret boyutunu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- işareti fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
```

```
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktasını taşı                                          |
//+------------------------------------------------------------------+
bool ArrowCheckMove(const long   chart_ID=0,        // çizelge kimliği
                    const string name="ArrowCheck", // nesne ismi
                    datetime     time=0,            // tutturma noktası zaman koordina
                    double       price=0)           // tutturma noktası fiyat koordina
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Kontrol tutturma noktası tipini değiştir                         |
//+------------------------------------------------------------------+
bool ArrowCheckAnchorChange(const long              chart_ID=0,        // çizelge kim
                            const string            name="ArrowCheck", // nesne ismi
                            const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // tutturma no
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktası tipini değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": tutturma noktası tipinin değiştirilmesi başarısız oldu! Hata kodu = ",G
      return(false);
     }
//--- başarılı çalıştırma
```

```
   return(true);
  }
//+------------------------------------------------------------------+
//| Kontrol işaretini sil                                            |
//+------------------------------------------------------------------+
bool ArrowCheckDelete(const long   chart_ID=0,        // çizelge kimliği
                      const string name="ArrowCheck") // işaret ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- işareti sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Kontrol\" işaretinin silinmesi başarısız! Hata kodu = ",GetLastError
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktası değerlerini kontrol et ve                       |
//| boş olanlar için varsayılan değerleri ayarla                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- noktanın zamanı ayarlanmamışsa, mevcut çubuk üzerinde olacak
   if(!time)
      time=TimeCurrent();
//--- noktanın fiyatı ayarlanmamışsa, Bid değerini alacak
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
```

```
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- işaretin çizileceği noktaları tanımla
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- Kontrol işaretini çizelge üzerinde oluştur
   if(!ArrowCheckCreate(0,InpName,0,date[d],price[p],InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, tutturma noktasını taşı ve konumunu işarete göre değiştir
//--- döngü sayacı
   int h_steps=bars*2/5;
//--- tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d<bars-1)
         d+=1;
      //--- tutturma noktasını taşı
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
```

```
      ChartRedraw();
      // 0.025 saniyelik gecikme
      Sleep(25);
     }
//--- tutturma noktasının konumunu işarete göre değiştir
   ArrowCheckAnchorChange(0,InpName,ANCHOR_BOTTOM);
//--- çizelgeyi yeniden çiz
   ChartRedraw();
//--- döngü sayacı
   h_steps=bars*2/5;
//--- tutturma noktasını taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(d<bars-1)
         d+=1;
      //--- tutturma noktasını taşı
      if(!ArrowCheckMove(0,InpName,date[d],price[p]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.025 saniyelik gecikme
      Sleep(25);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- işareti çizelgeden sil
   ArrowCheckDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_ARROW_LEFT_PRICE

Sol Fiyat Etiketi

Örnek

Aşağıdaki script, sol fiyat etiketini oluşturur ve onu çizelge üzerinde taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, sol fiyat etiketini çizelge üzerinde oluşturur."
#property description "tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string            InpName="LeftPrice";  // Fiyat etiketinin ismi
input int               InpDate=100;          // % şeklinde tutturma noktası tarih koo
input int               InpPrice=10;          // % şeklinde tutturma noktası fiyat koo
input color             InpColor=clrRed;      // Fiyat etiketi rengi
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // Kenar çizgisi stili
input int               InpWidth=2;           // Fiyat etiketi büyüklüğü
input bool              InpBack=false;        // Arkaplan etiketi
input bool              InpSelection=true;    // Taşıma için vurgula
input bool              InpHidden=true;       // Nesne listesinde gizle
input long              InpZOrder=0;          // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Sol fiyat etiketini oluştur                                      |
```

```
//+------------------------------------------------------------------+
bool ArrowLeftPriceCreate(const long            chart_ID=0,        // çizelge kimliği
                          const string          name="LeftPrice",  // fiyat etiketinin
                          const int             sub_window=0,      // alt-pencere indi
                          datetime              time=0,            // tutturma noktası
                          double                price=0,           // tutturma noktası
                          const color           clr=clrRed,        // fiyat etiketi re
                          const ENUM_LINE_STYLE style=STYLE_SOLID, // kenar çizgisi st
                          const int             width=1,           // fiyat etiketi bü
                          const bool            back=false,        // arka-planda göst
                          const bool            selection=true,    // taşıma için vurg
                          const bool            hidden=true,       // nesne listesinde
                          const long            z_order=0)         // fare tıklaması i
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeArrowEmptyPoint(time,price);
//--- hata değerini sıfırla
   ResetLastError();
//--- bir fiyat etiketi oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_LEFT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": sol fiyat etiketinin oluşturulması başarısız! Hata kodu = ",GetLastErro
      return(false);
     }
//--- etiket rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- kenar çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- etiket büyüklüğünü ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- etiketi fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktasını taşı                                          |
//+------------------------------------------------------------------+
```

```
bool ArrowLeftPriceMove(const long   chart_ID=0,       // çizelge kimliği
                        const string name="LeftPrice", // etiket ismi
                        datetime     time=0,           // tutturma noktasının zaman kı
                        double       price=0)          // tutturma noktasının fiyat kı
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Sol fiyat etiketini çizelgeden sil                               |
//+------------------------------------------------------------------+
bool ArrowLeftPriceDelete(const long   chart_ID=0,       // çizelge kimliği
                          const string name="LeftPrice") // etiket ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- etiketi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": sol fiyat etiketinin silinmesi başarısız oldu! Hata kodu = ",GetLastErr
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktası değerlerini kontrol et ve                       |
//| boş olanlar için varsayılan değerleri ayarla                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- noktanın zamanı ayarlanmamışsa, mevcut çubuk üzerinde olacak
   if(!time)
      time=TimeCurrent();
```

```
//--- noktanın fiyatı ayarlanmamışsa, Bid değerini alacak
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- etiketin çizileceği noktaları tanımla
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- sol fiyat etiketini çizelge üzerinde oluştur
   if(!ArrowLeftPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
```

```
      }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, tutturma noktasını taşı
//--- döngü sayacı
   int v_steps=accuracy*4/5;
//--- tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p<accuracy-1)
         p+=1;
      //--- tutturma noktasını taşı
      if(!ArrowLeftPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- etiketi çizelgeden sil
   ArrowLeftPriceDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_ARROW_RIGHT_PRICE

Sağ Fiyat Etiketi.

Örnek

Aşağıdaki script, sağ fiyat etiketini oluşturur ve onu çizelge üzerinde taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, sağ fiyat etiketini çizelge üzerinde oluşturur."
#property description "tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string            InpName="RightPrice"; // Fiyat etiketi ismi
input int               InpDate=0;            // % şeklinde tutturma noktası tarihi
input int               InpPrice=90;          // % şeklinde tutturma noktası fiyatı
input color             InpColor=clrRed;      // Fiyat etiketi rengi
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID; // Kenar çizgisi stili
input int               InpWidth=2;           // Fiyat etiketi büyüklüğü
input bool              InpBack=false;        // Arkaplan etiketi
input bool              InpSelection=true;    // Taşıma için vurgula
input bool              InpHidden=true;       // Nesne listesinde gizle
input long              InpZOrder=0;          // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Sağ fiyat etiketini oluştur                                      |
```

```
//+------------------------------------------------------------------+
bool ArrowRightPriceCreate(const long            chart_ID=0,        // çizelge tanımla
                           const string          name="RightPrice", // fiyat etiketi i
                           const int             sub_window=0,      // alt-pencere ind
                           datetime              time=0,            // tutturma noktas
                           double                price=0,           // tutturma noktas
                           const color           clr=clrRed,        // fiyat etiketi r
                           const ENUM_LINE_STYLE style=STYLE_SOLID, // kenar çizgisi s
                           const int             width=1,           // fiyat etiketi k
                           const bool            back=false,        // arkaplana al
                           const bool            selection=true,    // taşıma için vur
                           const bool            hidden=true,       // nesne listesind
                           const long            z_order=0)         // fare tıklaması
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeArrowEmptyPoint(time,price);
//--- hata değerini sıfırla
   ResetLastError();
//--- bir fiyat etiketi oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_RIGHT_PRICE,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": sağ fiyat etiketinin oluşturulması başarısız oldu! Hata kodu = ",GetLas
      return(false);
     }
//--- etiket rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- kenar çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- etiket büyüklüğünü ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- etiketi fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktasını taşı                                          |
//+------------------------------------------------------------------+
```

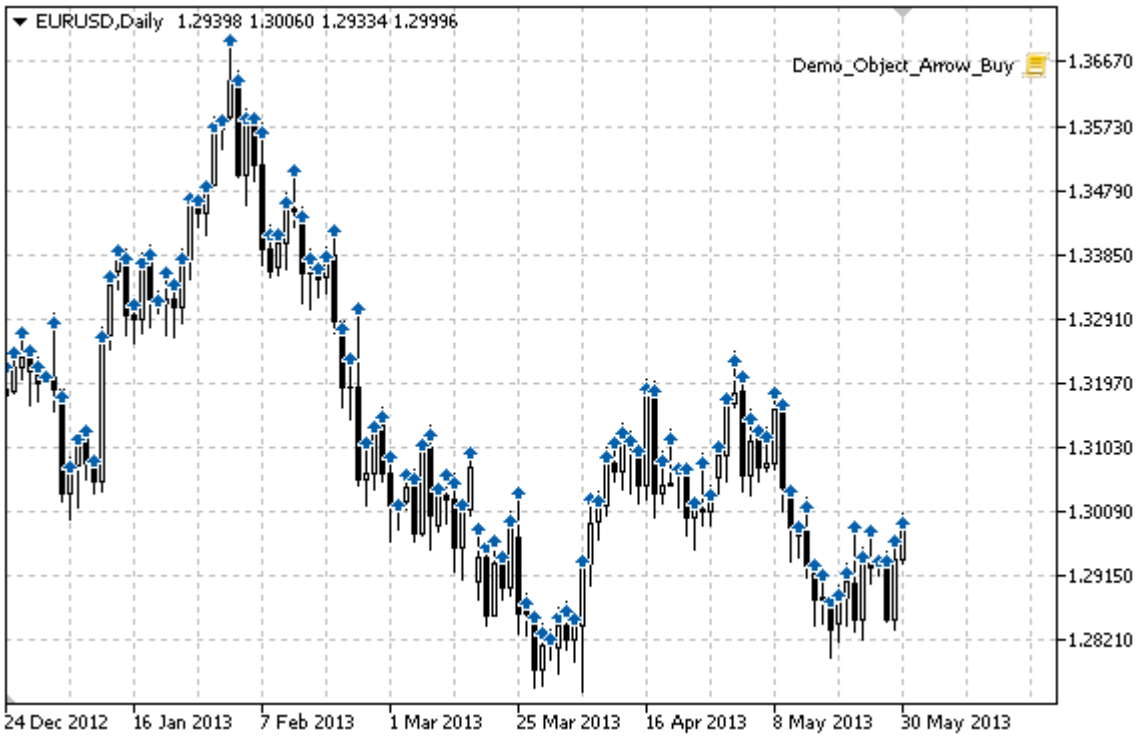
```
bool ArrowRightPriceMove(const long   chart_ID=0,        // çizelge kimliği
                         const string name="RightPrice", // etiket ismi
                         datetime     time=0,            // tutturma noktasının zaman
                         double       price=0)           // tutturma noktasının fiyat
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Sağ fiyat etiketini çizelgeden sil                               |
//+------------------------------------------------------------------+
bool ArrowRightPriceDelete(const long   chart_ID=0,        // çizelge kimliği
                           const string name="RightPrice") // etiket ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- etiketi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": sağ fiyat etiketinin silinmesi başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktası değerlerini kontrol et ve                       |
//| boş olanlar için varsayılan değerleri ayarla                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- noktanın zamanı ayarlanmamışsa, mevcut çubuk üzerinde olacak
   if(!time)
      time=TimeCurrent();
```

```
//--- noktanın fiyatı ayarlanmamışsa, Bid değerini alacak
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- etiketin çizileceği noktaları tanımla
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- sağ fiyat etiketini çizelge üzerinde oluştur
   if(!ArrowRightPriceCreate(0,InpName,0,date[d],price[p],InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
```

```
      }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, tutturma noktasını taşı
//--- döngü sayacı
   int v_steps=accuracy*4/5;
//--- tutturma noktasını taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- bir sonraki değeri kullan
      if(p>1)
         p-=1;
      //--- tutturma noktasını taşı
      if(!ArrowRightPriceMove(0,InpName,date[d],price[p]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- etiketi çizelgeden sil
   ArrowRightPriceDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_ARROW_BUY

Alış sinyali.



Örnek

Aşağıdaki script, Alış sinyalini oluşturur ve çizelge üzerinde taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Bu script, \"Alış\" sinyalini çizelge penceresine çizer."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input color InpColor=C'3,95,172'; // Işaretlerin rengi
//+------------------------------------------------------------------+
//| Alış sinyalini oluştur                                           |
//+------------------------------------------------------------------+
bool ArrowBuyCreate(const long            chart_ID=0,        // çizelge tanımlayıcısı
                    const string          name="ArrowBuy",   // işaret (sinyal) ismi
                    const int             sub_window=0,      // alt pencere indisi
                    datetime              time=0,            // tutturma noktası zaman
                    double                price=0,           // tutturma noktası fiyat
                    const color           clr=C'3,95,172',   // işaret rengi
                    const ENUM_LINE_STYLE style=STYLE_SOLID, // çizgi stili (vurgulanc
                    const int             width=1,           // çizgi boyutu (vurgular
                    const bool            back=false,        // arka planda
                    const bool            selection=false,   // taşıma için vurgula
```

```
                    const bool            hidden=true,       // nesneyi listede gizle
                    const long            z_order=0)         // fare tıklaması için ön
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeArrowEmptyPoint(time,price);
//--- hata değerini sıfırla
   ResetLastError();
//--- işareti oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_BUY,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"Alım\" sinyali oluşturma başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- bir işaret rengi ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- bir çizgi stili ayarla (vurgulandığında)
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- bir çizgi boyutu ayarla (vurgulandığında)
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- işareti fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktasını taşı                                          |
//+------------------------------------------------------------------+
bool ArrowBuyMove(const long   chart_ID=0,       // çizelge kimliği
                  const string name="ArrowBuy", // nesne ismi
                  datetime     time=0,          // tutturma noktasının zaman koordinat
                  double       price=0)         // tutturma noktasının fiyat koordinat
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,0,time,price))
```

```
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Alış sinyalini sil                                               |
//+------------------------------------------------------------------+
bool ArrowBuyDelete(const long   chart_ID=0,      // çizelge tanımlayıcısı
                    const string name="ArrowBuy") // işaret (sinyal) ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- işareti sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Alış\" işaretinin silinmesi başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktası değerlerini kontrol et ve                       |
//| boş olanlar için varsayılan değerleri ayarla                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- noktanın zamanı ayarlanmamışsa, mevcut çubuk üzerinde olacak
   if(!time)
      time=TimeCurrent();
//--- noktanın fiyatı ayarlanmamışsa, Bid değerini alacak
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // görünür çubukların tarihlerini depolamak için bir dizi
   double   low[];  // görünür çubukların Low (düşük) fiyatlarını depolamak için bir d
   double   high[]; // görünür çubukların High (yüksek) fiyatlarını depolamak için bi
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

```
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- Low (düşük) fiyatların dizisini doldur
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("Low (düşük) fiyat değerlerinin kopyalanması başarısız oldu! Hata kodu = "
      return;
     }
//--- High (yüksek) fiyat dizisini doldur
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("High (yüksek) fiyat değerlerinin kopyalanması başarısız oldu! Hata kodu =
      return;
     }
//--- Her bir görünür çubuk için düşük noktalarda Alış işaretleri oluştur
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyCreate(0,"ArrowBuy_"+(string)i,0,date[i],low[i],InpColor))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- Alış işaretini her görünür çubuk için High (yüksek) noktaya taşı
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyMove(0,"ArrowBuy_"+(string)i,date[i],high[i]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
```

```
//--- Alış işaretini sil
   for(int i=0;i<bars;i++)
     {
      if(!ArrowBuyDelete(0,"ArrowBuy_"+(string)i))
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//---
  }
```

OBJ_ARROW_SELL

Satış işareti.

Örnek

Aşağıdaki betik, Satış işaretini çizelge üzerinde oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script \"Satış\" işaretini çizelge penceresinde oluşturur."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input color InpColor=C'225,68,29'; // İşaretlerin rengi
//+------------------------------------------------------------------+
//| Satış işaretini oluştur                                          |
//+------------------------------------------------------------------+
bool ArrowSellCreate(const long            chart_ID=0,        // çizelge kimliği
                     const string          name="ArrowSell",  // işaretin ismi
                     const int             sub_window=0,      // alt-pencere indisi
                     datetime              time=0,            // tutturma noktası zama
                     double                price=0,           // tutturma noktası fiya
                     const color           clr=C'225,68,29',  // işaret rengi
                     const ENUM_LINE_STYLE style=STYLE_SOLID, // çizgi stili (vurgular
                     const int             width=1,           // çizgi kalınlığı (vurg
                     const bool            back=false,        // arka-plan çizimi
                     const bool            selection=false,   // taşıma için vurgula
```

```
                     const bool            hidden=true,       // nesne listesinde gizl
                     const long            z_order=0)         // fare tıklaması için ö
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeArrowEmptyPoint(time,price);
//--- hata değerini sıfırla
   ResetLastError();
//--- işareti oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW_SELL,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"Satış\" işaretinin oluşturulması başarısız! Hata kodu = ",GetLastErro
      return(false);
     }
//--- bir işaret rengi ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- bir çizgi stili ayarla (vurgulandığında)
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- bir çizgi boyutu ayarla (vurgulandığında)
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- işareti fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktasını taşı                                          |
//+------------------------------------------------------------------+
bool ArrowSellMove(const long   chart_ID=0,       // çizelge kimliği
                   const string name="ArrowSell", // nesne ismi
                   datetime     time=0,           // tutturma noktasının zaman koordi
                   double       price=0)          // tutturma noktası fiyat koordinat
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,0,time,price))
```

```
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Satış işaretini sil                                              |
//+------------------------------------------------------------------+
bool ArrowSellDelete(const long   chart_ID=0,       // çizelge kimliği
                     const string name="ArrowSell") // işaret ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- işareti sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Satış\" işaretinin silinmesi başarısız oldu! Hata kodu = ",GetLastEr
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktası değerlerini kontrol et ve                       |
//| boş olanlar için varsayılan değerleri ayarla                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
  {
//--- noktanın zamanı ayarlanmamışsa, mevcut çubuk üzerinde olacak
   if(!time)
      time=TimeCurrent();
//--- noktanın fiyatı ayarlanmamışsa, Bid değerini alacak
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // görünür çubukların tarihlerini depolamak için bir dizi
   double   low[];  // görünür çubukların Low (düşük) fiyatlarını depolamak için bir
   double   high[]; // görünür çubukların High (yüksek) fiyatlarını depolamak için bi
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
```

```
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- Low (düşük) fiyatların dizisini doldur
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("Low (düşük) fiyat değerlerinin kopyalanması başarısız oldu! Hata kodu = "
      return;
     }
//--- High (yüksek) fiyat dizisini doldur
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("High (yüksek) fiyat değerlerinin kopyalanması başarısız oldu! Hata kodu =
      return;
     }
//--- her görünür çubuk için High (yüksek fiyat) noktalarında Satış işaretini oluştur
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellCreate(0,"ArrowSell_"+(string)i,0,date[i],high[i],InpColor))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- satış işaretlerini, her görünür çubuk için Low (düşük fiyat) noktalarına taşı
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellMove(0,"ArrowSell_"+(string)i,date[i],low[i]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
```

```
//--- satış işaretlerini sil
   for(int i=0;i<bars;i++)
     {
      if(!ArrowSellDelete(0,"ArrowSell_"+(string)i))
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//---
  }
```

OBJ_ARROW

Ok nesnesi.

Not

Nesneye göre tutturma noktası konumu ENUM_ARROW_ANCHOR sayımından seçilebilir.

MetaEditor'de bir kod yazarken büyük oklar (5'ten büyük), sadece uygun OBJPROP_WIDTH özellik değerini ayarlayarak oluşturulabilir.

Gerekli ok tipi Wingdings yazı tipinin sembol kodlarından birinin ayarlanmasıyla seçilebilir.

Örnek

Aşağıdaki script, Ok nesnesini oluşturur ve onu çizelge üzerinde taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, çizelge üzerinde bir rassal ok nesnesi oluşturur."
#property description "tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string             InpName="Arrow";         // Ok ismi
input int                InpDate=50;              // % şeklinde tutturma (çapa) noktası
input int                InpPrice=50;             // % şeklinde tutturma noktası fiyatı
input ENUM_ARROW_ANCHOR  InpAnchor=ANCHOR_TOP;    // Çapa tipi
```

```
input color             InpColor=clrDodgerBlue; // Ok rengi
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID;   // Kenar çizgisi stili
input int               InpWidth=10;            // Ok boyutu
input bool              InpBack=false;          // Ok arka-planda
input bool              InpSelection=false;     // Taşıma için vurgula
input bool              InpHidden=true;         // Nesne listesinde gizle
input long              InpZOrder=0;            // Fare tıklaması önceliği
//+------------------------------------------------------------------+
//| Oku oluştur                                                      |
//+------------------------------------------------------------------+
bool ArrowCreate(const long              chart_ID=0,           // çizelge kimliği
                 const string            name="Arrow",         // ok ismi
                 const int               sub_window=0,         // alt-pencere indisi
                 datetime                time=0,               // tutturma (çapa) nokt
                 double                  price=0,              // tutturma noktası fiy
                 const uchar             arrow_code=252,       // ok kodu
                 const ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM, // tutturma noktası kor
                 const color             clr=clrRed,           // ok rengi
                 const ENUM_LINE_STYLE   style=STYLE_SOLID,    // kenar çizgisi stili
                 const int               width=3,              // ok boyutu
                 const bool              back=false,           // arka-planda
                 const bool              selection=true,       // taşıma için vurgula
                 const bool              hidden=true,          // nesne listesinde giz
                 const long              z_order=0)            // fare tıklaması için
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeArrowEmptyPoint(time,price);
//--- hata değerini sıfırla
   ResetLastError();
//--- bir ok oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_ARROW,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": okun oluşturulması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- ok kodunu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,arrow_code);
//--- tutturma noktası tipini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- ok rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- kenar çizgi stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- ok boyutunu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- oku fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
```

```
//--- ObjectCreate fonksiyonu ile bir nesne oluşturulurken,
//--- nesne ön tanımlı olarak vurgulanamaz veya taşınamaz. Bu yöntemin içinde, paramet
//--- seçilmesi, nesnenin vurgulanmasını ve taşınmasını mümkün kılar
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktasını taşı                                          |
//+------------------------------------------------------------------+
bool ArrowMove(const long   chart_ID=0,   // çizelge kimliği
               const string name="Arrow", // nesne ismi
               datetime     time=0,       // tutturma noktası zaman koordinatı
               double       price=0)      // tutturma noktası fiyat koordinatı
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Ok kodunu değiştir                                               |
//+------------------------------------------------------------------+
bool ArrowCodeChange(const long   chart_ID=0,   // çizelge kimliği
                     const string name="Arrow", // nesne ismi
                     const uchar  code=252)     // ok kodu
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- ok kodunu değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ARROWCODE,code))
     {
```

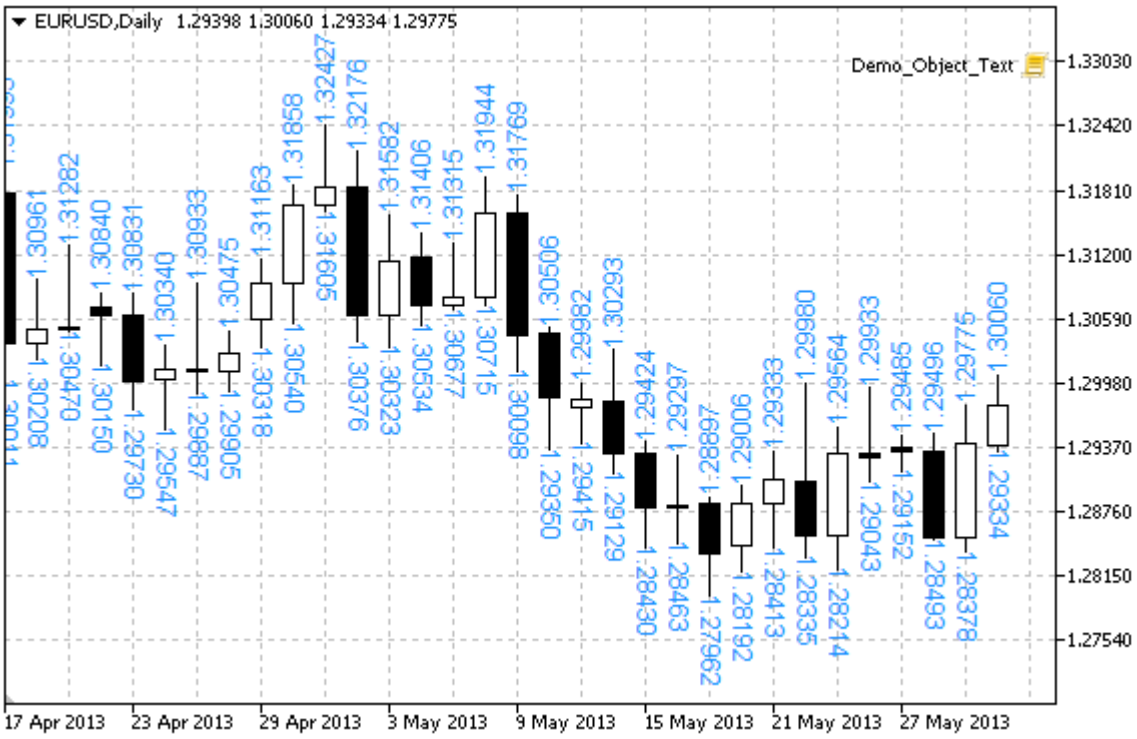
```
      Print(__FUNCTION__,
            ": ok kodunun değiştirilmesi başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Çapa tipini değiştir                                             |
//+------------------------------------------------------------------+
bool ArrowAnchorChange(const long              chart_ID=0,        // çizelge kimliği
                       const string            name="Arrow",      // nesne ismi
                       const ENUM_ARROW_ANCHOR anchor=ANCHOR_TOP) // çapa tipi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktası tipini değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor))
     {
      Print(__FUNCTION__,
            ": tutturma noktası tipinin değiştirilmesi başarısız oldu! Hata kodu = ",G
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Oku sil                                                          |
//+------------------------------------------------------------------+
bool ArrowDelete(const long   chart_ID=0,   // çizelge kimliği
                 const string name="Arrow") // ok ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- oku sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": okun silinmesi başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktası değerlerini kontrol et ve                       |
//| boş olanlar için varsayılan değerleri ayarla                     |
//+------------------------------------------------------------------+
void ChangeArrowEmptyPoint(datetime &time,double &price)
```

```
  {
//--- noktanın zamanı ayarlanmamışsa, mevcut çubuk üzerinde olacak
   if(!time)
      time=TimeCurrent();
//--- noktanın fiyatı ayarlanmamışsa, Bid değerini alacak
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate<0 || InpDate>100 || InpPrice<0 || InpPrice>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- fiyat dizisi büyüklüğü
   int accuracy=1000;
//--- nesnenin tutturma noktası konumunu değiştirmek için kullanılacak
//--- tarih ve fiyat değerlerinin saklanacağı diziler
   datetime date[];
   double   price[];
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(price,accuracy);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- fiyat dizisini değerlerle doldur
//--- çizelgedeki en yüksek ve en düşük değerleri bul
   double max_price=ChartGetDouble(0,CHART_PRICE_MAX);
   double min_price=ChartGetDouble(0,CHART_PRICE_MIN);
//--- fiyatlar için bir değişim birimini tanımla ve diziyi doldur
   double step=(max_price-min_price)/accuracy;
   for(int i=0;i<accuracy;i++)
      price[i]=min_price+i*step;
//--- okun çizileceği noktaları tanımla
   int d=InpDate*(bars-1)/100;
   int p=InpPrice*(accuracy-1)/100;
//--- çizelge üzerinde bir ok oluştur
```

```
   if(!ArrowCreate(0,InpName,0,date[d],price[p],32,InpAnchor,InpColor,
      InpStyle,InpWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yeniden çiz
   ChartRedraw();
//--- döngü içinde tüm ok oluşturma durumlarını göz önüne al
   for(int i=33;i<256;i++)
     {
      if(!ArrowCodeChange(0,InpName,(uchar)i))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // yarım saniyelik gecikme
      Sleep(500);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- oku çizelgeden sil
   ArrowDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_TEXT

Metin nesnesi.



Not

Tutturma noktası konumu, metne göre ENUM_ANCHOR_POINT sayımından seçilebilir. Ayrıca, OBJPROP_ANGLE özelliğini kullanarak metnin eğim açısını da değiştirebilirsiniz

Örnek

Aşağıdaki script, çizelge üzerinde metin nesneleri oluşturur. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script creates \"Text\" grafiksel nesneler."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string            InpFont="Arial";         // Yazı tipi
input int               InpFontSize=10;          // Yazı tipi boyutu
input color             InpColor=clrRed;         // Renk
input double            InpAngle=90.0;           // Derece olarak eğim açısı
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_LEFT;   // Tutturma (çapa) tipi
input bool              InpBack=false;           // Arkaplan nesnesi
input bool              InpSelection=false;      // Taşıma için vurgula
input bool              InpHidden=true;          // Nesne listesinde gizleme
input long              InpZOrder=0;             // Fare tıklaması için öncelik
```

```
//+------------------------------------------------------------------+
//| Metin nesnesinin oluşturulması                                   |
//+------------------------------------------------------------------+
bool TextCreate(const long              chart_ID=0,               // çizelge kimliği
                const string            name="Text",              // nesne ismi
                const int               sub_window=0,             // alt-pencere indis
                datetime                time=0,                   // tutturma (çapa) n
                double                  price=0,                  // tutturma noktası
                const string            text="Text",              // metnin kendisi
                const string            font="Arial",             // yazı tipi
                const int               font_size=10,             // yazı tipi boyutu
                const color             clr=clrRed,               // renk
                const double            angle=0.0,                // metin eğimi
                const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // çapa tipi
                const bool              back=false,               // arka-planda
                const bool              selection=false,          // taşıma için vurgu
                const bool              hidden=true,              // nesne listesinde
                const long              z_order=0)                // fare tıklaması iç
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeTextEmptyPoint(time,price);
//--- hata değerini sıfırla
   ResetLastError();
//--- Metin nesnesini olştur
   if(!ObjectCreate(chart_ID,name,OBJ_TEXT,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": \"Metin\" nesnesinin oluşturulması başarısız oldu! Hata kodu = ",GetLas
      return(false);
     }
//--- metni ayarla
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- metin yazı tipini ayarla
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- yazı tipi boyutunu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- metnin eğim açısını ayarla
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- tutturma noktası tipini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- rengi ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- nesneyi fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
```

```
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktasını taşı                                          |
//+------------------------------------------------------------------+
bool TextMove(const long   chart_ID=0,  // çizelge kimliği
              const string name="Text", // nesne ismi
              datetime     time=0,      // tutturma noktasının zaman koordinatı
              double       price=0)     // tutturma noktasının fiyat koordinatı
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğa
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Nesnenin metnini değiştir                                        |
//+------------------------------------------------------------------+
bool TextChange(const long   chart_ID=0,  // çizelge tanımlayıcı
                const string name="Text", // nesne ismi
                const string text="Text") // metin
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesne metnini değiştir
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": metin değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
```

```
//+------------------------------------------------------------------+
//| Metin nesnesini sil                                              |
//+------------------------------------------------------------------+
bool TextDelete(const long   chart_ID=0,  // çizelge kimliği
                const string name="Text") // nesne ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesneyi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Metin\" nesnesi silinemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktası değerlerini kontrol et ve                       |
//| boş olanlar için varsayılan değerleri ayarla                     |
//+------------------------------------------------------------------+
void ChangeTextEmptyPoint(datetime &time,double &price)
  {
//--- noktanın zamanı ayarlanmamışsa, mevcut çubuk üzerinde olacak
   if(!time)
      time=TimeCurrent();
//--- noktanın fiyatı ayarlanmamışsa, Bid değerini alacak
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[]; // görünür çubukların tarihlerini depolamak için bir dizi
   double   low[];  // görünür çubukların Low (düşük) fiyatlarını depolamak için bir d
   double   high[]; // görünür çubukların High (yüksek) fiyatlarını depolamak için bi
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(low,bars);
   ArrayResize(high,bars);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
```

```
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- Low (düşük) fiyatların dizisini doldur
   if(CopyLow(Symbol(),Period(),0,bars,low)==-1)
     {
      Print("Low (düşük) fiyat değerlerinin kopyalanması başarısız oldu! Hata kodu = "
      return;
     }
//--- High (yüksek) fiyat dizisini doldur
   if(CopyHigh(Symbol(),Period(),0,bars,high)==-1)
     {
      Print("High (yüksek) fiyat değerlerinin kopyalanması başarısız oldu! Hata kodu =
      return;
     }
//--- metinler ne sıklıkta gösterilecek, tanımla
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- adım değerini taımla
   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;
     }
//--- High (yüksek) ve Low (düşük) çubukların değerleri (boşluklu olarak) için metinle
   for(int i=0;i<bars;i+=step)
     {
      //--- metinleri oluştur
      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;
        }
      //--- script işlemi devre dışı bırakıldı mı kontrol et
```

```
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- yarım saniyelik gecikme
   Sleep(500);
//--- metinleri sil
   for(int i=0;i<bars;i+=step)
     {
      if(!TextDelete(0,"TextHigh_"+(string)i))
         return;
      if(!TextDelete(0,"TextLow_"+(string)i))
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//---
  }
```

OBJ_LABEL

Etiket nesnesi.

Not

İşarete göre tutturma noktası (çapa) konumu ENUM_ARROW_ANCHOR sayımından seçilebilir. Tutturma noktası koordinatları piksel bazında ayarlanır.

Ayrıca ENUM_BASE_CORNER sayımını kullanarak, metin etiketinin tutturma (çapa) köşesini de seçebilirsiniz.

Örnek

Aşağıdaki script, "Düzenle" nesnesini oluşturur ve çizelge üzerinde taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Etiket\" grafikksel nesnesini oluşturur."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string            InpName="Label";          // Etiket ismi
input int               InpX=150;                 // X-ekseni uzunluğu
input int               InpY=150;                 // Y-ekseni uzunluğu
input string            InpFont="Arial";          // Yazı tipi
input int               InpFontSize=14;           // Yazı tipi büyüklüğü
input color             InpColor=clrRed;          // Renk
input double            InpAngle=0.0;             // Derece olarak eğim açısı
```

```
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER; // Tutturma (çapa) tipi
input bool              InpBack=false;           // Arkaplan nesnesi
input bool              InpSelection=true;       // Taşıma için vurgula
input bool              InpHidden=true;          // Nesne listesinde gizleme
input long              InpZOrder=0;             // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Bir metin etiketi oluştur                                        |
//+------------------------------------------------------------------+
bool LabelCreate(const long              chart_ID=0,               // çizelge kimliği
                 const string            name="Label",             // etiket ismi
                 const int               sub_window=0,             // alt pencere indi
                 const int               x=0,                      // X koordinatı
                 const int               y=0,                      // Y koordinatı
                 const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // tutturma yapılac
                 const string            text="Label",             // metin
                 const string            font="Arial",             // yazı tipi
                 const int               font_size=10,             // yazı tipi boyutu
                 const color             clr=clrRed,               // renk
                 const double            angle=0.0,                // metin eğimi
                 const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // çapa tipi
                 const bool              back=false,               // arka-planda
                 const bool              selection=false,          // taşımak için vur
                 const bool              hidden=true,              // nesne listesinde
                 const long              z_order=0)                // fare tıklaması ç
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- metin etiketi oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": metin etiketinin oluşturulması başarısız oldu! Hata kodu = ",GetLastErr
      return(false);
     }
//--- etiket koordinatlarını ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- tanımlanan nokta koordinatlarına göre çapa (tutturma) köşesini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- metni ayarla
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- metin yazı tipini ayarla
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- yazı tipi boyutunu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- metnin eğim açısını ayarla
   ObjectSetDouble(chart_ID,name,OBJPROP_ANGLE,angle);
//--- tutturma noktası tipini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
```

```
//--- rengi ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- etiketi fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Metin etiketini taşı                                             |
//+------------------------------------------------------------------+
bool LabelMove(const long   chart_ID=0,   // çizelge kimliği
               const string name="Label", // etiket ismi
               const int    x=0,          // X koordinatı
               const int    y=0)          // Y koordinatı
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- metin etiketini taşı
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": etiketin X koordinatı taşınamadı! Hata kodu = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": etiketin Y koordinatı taşınamadı! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Etiketin bağlanacağı çizelge köşesini değiştir                   |
//+------------------------------------------------------------------+
bool LabelChangeCorner(const long             chart_ID=0,               // çizelge kir
                       const string           name="Label",             // etiket ismi
                       const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // tutturma ya
  {
//--- hata değerini sıfırla
   ResetLastError();
```

```
//--- tutturma köşesini değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": tutturma köşesi değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Nesnenin metnini değiştir                                        |
//+------------------------------------------------------------------+
bool LabelTextChange(const long   chart_ID=0,   // çizelge kimliği
                     const string name="Label", // nesne ismi
                     const string text="Text")  // metint
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesne metnini değiştir
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": metin değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Metin etiketini sil                                              |
//+------------------------------------------------------------------+
bool LabelDelete(const long   chart_ID=0,   // çizelge kimliği
                 const string name="Label") // etiket ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- etiketi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": metin etiketini silinemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- etiketin koordinatlarını yerel değişkene depola
   int x=InpX;
   int y=InpY;
//--- çizelge penceresi büyüklüğü
   long x_distance;
   long y_distance;
//--- pencere büyüklüğünü ayarla
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Çizelgenin genişlik değeri alınamadı! Hata kodu = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Çizelgenin yükseklik değeri alınamadı! Hata kodu = ",GetLastError());
      return;
     }
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpX<0 || InpX>x_distance-1 || InpY<0 || InpY>y_distance-1)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- etiket için başlangıç metnini hazırla
   string text;
   StringConcatenate(text,"Sol üst köşe: ",x,",",y);
//--- çizelge üzerinde bir metin etiketi oluştur
   if(!LabelCreate(0,InpName,0,InpX,InpY,CORNER_LEFT_UPPER,text,InpFont,InpFontSize,
      InpColor,InpAngle,InpAnchor,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve yarım saniye bekle
   ChartRedraw();
   Sleep(500);
//--- etiketi taşı ve aynı anda metnini de değiştir
//--- eksen üzerindeki tekrarlama sayısı
   int h_steps=(int)(x_distance/2-InpX);
   int v_steps=(int)(y_distance/2-InpY);
//--- etiketi aşağı taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- koordinatı değiştir
      y+=2;
      //--- etiketi taşı ve metnini değiştir
      MoveAndTextChange(x,y,"Sol üst köşe: ");
```

```
      }
//--- yarım saniyelik gecikme
   Sleep(500);
//--- etiketi sağa taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- koordinatı değiştir
      x+=2;
      //--- etiketi taşı ve metnini değiştir
      MoveAndTextChange(x,y,"Sol üst köşe: ");
     }
//--- yarım saniyelik gecikme
   Sleep(500);
//--- etiketi yukarı taşı
   for(int i=0;i<v_steps;i++)
     {
      //--- koordinatı değiştir
      y-=2;
      //--- etiketi taşı ve metnini değiştir
      MoveAndTextChange(x,y,"Sol üst köşe: ");
     }
//--- yarım saniyelik gecikme
   Sleep(500);
//--- etiketi sola taşı
   for(int i=0;i<h_steps;i++)
     {
      //--- koordinatı değiştir
      x-=2;
      //--- etiketi taşı ve metnini değiştir
      MoveAndTextChange(x,y,"Sol üst köşe: ");
     }
//--- yarım saniyelik gecikme
   Sleep(500);
//--- şimdi, tutturma köşesini değiştirerek noktayı taşı
//--- sol alt köşeye taşı
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_LOWER))
      return;
//--- etiket metnini değiştir
   StringConcatenate(text,"Sol alt köşe: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- çizelgeyi yeniden çiz ve iki saniye bekle
   ChartRedraw();
   Sleep(2000);
//--- sağ alt köşeye taşı
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_LOWER))
      return;
//--- etiket metnini değiştir
   StringConcatenate(text,"Sol alt köşe: ",x,",",y);
```

```
   if(!LabelTextChange(0,InpName,text))
      return;
//--- çizelgeyi yeniden çiz ve iki saniye bekle
   ChartRedraw();
   Sleep(2000);
//--- sağ üst köşeye taşı
   if(!LabelChangeCorner(0,InpName,CORNER_RIGHT_UPPER))
      return;
//--- etiket metnini değiştir
   StringConcatenate(text,"Sağ üst köşe: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- çizelgeyi yeniden çiz ve iki saniye bekle
   ChartRedraw();
   Sleep(2000);
//--- sol üst köşeye taşı
   if(!LabelChangeCorner(0,InpName,CORNER_LEFT_UPPER))
      return;
//--- etiket metnini değiştir
   StringConcatenate(text,"Sol üst köşe: ",x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return;
//--- çizelgeyi yeniden çiz ve iki saniye bekle
   ChartRedraw();
   Sleep(2000);
//--- etiketi sil
   LabelDelete(0,InpName);
//--- çizelgeyi yenile ve yarım saniye bekle
   ChartRedraw();
   Sleep(500);
//---
  }
//+------------------------------------------------------------------+
//| Fonksiyon, nesneyi taşır ve metnini değiştirir                   |
//+------------------------------------------------------------------+
bool MoveAndTextChange(const int x,const int y,string text)
  {
//--- etiketi taşı
   if(!LabelMove(0,InpName,x,y))
      return(false);
//--- etiket metnini değiştir
   StringConcatenate(text,text,x,",",y);
   if(!LabelTextChange(0,InpName,text))
      return(false);
//--- script işlemi zorla sonlandırılmış mı kontrol et
   if(IsStopped())
      return(false);
//--- çizelgeyi yeniden çiz
   ChartRedraw();
```

```
// 0.01 saniyelik gecikme
   Sleep(10);
//--- fonksiyondan çık
   return(true);
  }
```

OBJ_BUTTON

Düğme nesnesi.

Not

Tutturma noktası koordinatları piksel bazında ayarlanır. Düğmenin tutturma köşesini ENUM_BASE_CORNER sayımından seçebilirsiniz.

Örnek

Aşağıdaki script, düğme nesnesini oluşturur ve çizelge üzerinde taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, çizelge üzerinde bir düğme oluşturur."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string           InpName="Button";            // Düğme ismi
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Tutturma için çizelge köşesi
input string           InpFont="Arial";             // Yazı tipi
input int              InpFontSize=14;              // Yazı tipi büyüklüğü
input color            InpColor=clrBlack;           // Metin rengi
input color            InpBackColor=C'236,233,216'; // Arka-plan rengi
input color            InpBorderColor=clrNONE;      // Kenar rengi
input bool             InpState=false;              // Basılı/Serbest
input bool             InpBack=false;               // Arka-plan nesnesi
input bool             InpSelection=false;          // Taşıma için vurgula
```

```
input bool              InpHidden=true;              // Nesne listesinde gizle
input long              InpZOrder=0;                 // Fare tıklaması önceliği
//+------------------------------------------------------------------+
//| Düğmeyi oluştur                                                  |
//+------------------------------------------------------------------+
bool ButtonCreate(const long              chart_ID=0,               // çizelge kimliği
                  const string            name="Button",            // düğme ismi
                  const int               sub_window=0,             // alt pencere ind
                  const int               x=0,                      // X koordinatı
                  const int               y=0,                      // Y koordinatı
                  const int               width=50,                 // düğme genişliği
                  const int               height=18,                // düğme yüksekliğ
                  const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // tutturulacak çi
                  const string            text="Button",            // metin
                  const string            font="Arial",             // yazı tipi
                  const int               font_size=10,             // yazı tipi boyut
                  const color             clr=clrBlack,             // metin rengi
                  const color             back_clr=C'236,233,216',  // arka-plan rengi
                  const color             border_clr=clrNONE,       // kenar rengi
                  const bool              state=false,              // basılı serbest
                  const bool              back=false,               // arkaplanda
                  const bool              selection=false,          // taşıma için vur
                  const bool              hidden=true,              // nesne listesind
                  const long              z_order=0)                // fare tıklaması
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- düğmeyi oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_BUTTON,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": düğmenin oluşturulması başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- düğme koordinatlarını ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- düğme boyutunu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- tanımlanan nokta koordinatlarına göre çapa (tutturma) köşesini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- metni ayarla
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- metin yazı tipini ayarla
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
//--- yazı tipi boyutunu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- metin rengini ayara
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- arkaplan rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- kenar çizgisinin rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- set button state
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- düğmeyi fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Düğmeyi taşı                                                     |
//+------------------------------------------------------------------+
bool ButtonMove(const long   chart_ID=0,    // çizelge kimliği
                const string name="Button", // düğme ismi
                const int    x=0,           // X koordinatı
                const int    y=0)           // Y koordinatı
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- düğmeyi taşı
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": düğmenin X koordinatı taşınamadı! Hata kodu = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": düğmenin Y koordinatı taşınamadı! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Düğme boyutunu değiştir                                          |
//+------------------------------------------------------------------+
bool ButtonChangeSize(const long   chart_ID=0,    // çizelge kimliği
```

```
                      const string name="Button", // düğme ismi
                      const int    width=50,      // düğme genişliği
                      const int    height=18)     // düğme yüksekliği
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- düğme boyutunu değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": düğme genişliği değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": düğme yüksekliği değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Düğmenin bağlanacağı çizelge köşesini değiştir                   |
//+------------------------------------------------------------------+
bool ButtonChangeCorner(const long             chart_ID=0,               // çizelge ki
                        const string           name="Button",            // düğme ismi
                        const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER) // tutturmani
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma köşesini değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner))
     {
      Print(__FUNCTION__,
            ": tutturma köşesi değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Düğme metnini değiştir                                           |
//+------------------------------------------------------------------+
bool ButtonTextChange(const long   chart_ID=0,    // çizelge kimliği
                      const string name="Button", // düğme ismi
                      const string text="Text")   // metin
  {
//--- hata değerini sıfırla
```

```
   ResetLastError();
//--- nesne metnini değiştir
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": metin değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Düğmeyi sil                                                      |
//+------------------------------------------------------------------+
bool ButtonDelete(const long   chart_ID=0,    // çizelge kimliği
                  const string name="Button") // düğme ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- düğmeyi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": düğme silinemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- çizelge penceresi büyüklüğü
   long x_distance;
   long y_distance;
//--- pencere büyüklüğünü ayarla
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Çizelgenin genişlik değeri alınamadı! Hata kodu = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Çizelgenin yükseklik değeri alınamadı! Hata kodu = ",GetLastError());
      return;
     }
//--- düğme boyutunun değiştirilmesi için adım değerini tanımla
```

```
   int x_step=(int)x_distance/32;
   int y_step=(int)y_distance/32;
//--- düğmenin koordinatlarını ve boyutunu ayarla
   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;
//--- düğmeyi oluştur
   if(!ButtonCreate(0,InpName,0,x,y,x_size,y_size,InpCorner,"Press",InpFont,InpFontSiz
      InpColor,InpBackColor,InpBorderColor,InpState,InpBack,InpSelection,InpHidden,Inp
     {
      return;
     }
//--- çizelgeyi yeniden çiz
   ChartRedraw();
//--- düğmeyi döngü içinde indirge
   int i=0;
   while(i<13)
     {
      //--- yarım saniyelik gecikme
      Sleep(500);
      //--- düğmeyi basılı konuma getir
      ObjectSetInteger(0,InpName,OBJPROP_STATE,true);
      //--- çizelgeyi yenile ve 0.2 saniye bekle
      ChartRedraw();
      Sleep(200);
      //--- koordinatları ve düğme büyüklüğünü yeniden tanımla
      x+=x_step;
      y+=y_step;
      x_size-=x_step*2;
      y_size-=y_step*2;
      //--- düğmeyi indirge
      ButtonMove(0,InpName,x,y);
      ButtonChangeSize(0,InpName,x_size,y_size);
      //--- düğmeyi serbest duruma getir
      ObjectSetInteger(0,InpName,OBJPROP_STATE,false);
      //--- çizelgeyi yenile
      ChartRedraw();
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- döngü sayacını artır
      i++;
     }
//--- yarım saniyelik gecikme
   Sleep(500);
//--- düğmeyi sil
   ButtonDelete(0,InpName);
   ChartRedraw();
```

```
//--- 1 saniye bekle
   Sleep(1000);
//---
  }
```

OBJ_CHART

Çizelge nesnesi.

Not

Görsel test sırasında "OBJ_CHART" türü nesneler desteklenmez (görüntülenmez).

Tutturma noktası koordinatları piksel bazında ayarlanır. Tutturma noktasını ENUM_BASE_CORNER sayımından seçebilirsiniz.

Çizelge nesnesi için sembol, periyot ve ölçek değerleri seçilebilir. Fiyat ölçeği ve tarih görüntüleme modu da ayrıca etkinleştirilebilir/devre dışı bırakılabilir.

Örnek

Aşağıdaki script, Çizelge nesnesini oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Çizelge nesnesini oluşturur\" nesnesini oluşturur."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string           InpName="Chart";              // Nesne ismi
input string           InpSymbol="EURUSD";           // Sembol
input ENUM_TIMEFRAMES  InpPeriod=PERIOD_H1;          // Periyot
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Tutturma köşesi
input int              InpScale=2;                   // Ölçek
```

```
input bool              InpDateScale=true;           // Zaman ölçeğinin görüntülenmesi
input bool              InpPriceScale=true;          // Fiyat ölçeğinin görüntülenmesi
input color             InpColor=clrRed;             // Kenar çizgisinin rengi (vurgula
input ENUM_LINE_STYLE   InpStyle=STYLE_DASHDOTDOT;   // Çizgi stili (vurgulandığında)
input int               InpPointWidth=1;             // Taşıma noktası boyutu
input bool              InpBack=false;               // Arka-plan nesnesi
input bool              InpSelection=true;           // Taşıma için vurgula
input bool              InpHidden=true;              // Nesne listesinde gizle
input long              InpZOrder=0;                 // Fare tıklaması önceliği
//+------------------------------------------------------------------+
//| Çizelge nesnesinin oluşturulması                                 |
//+------------------------------------------------------------------+
bool ObjectChartCreate(const long              chart_ID=0,               // çizelge k
                       const string            name="Chart",             // nesne ismi
                       const int               sub_window=0,             // alt-pence
                       const string            symbol="EURUSD",          // sembol
                       const ENUM_TIMEFRAMES   period=PERIOD_H1,         // periyot
                       const int               x=0,                      // X koordina
                       const int               y=0,                      // Y koordina
                       const int               width=300,                // genişlik
                       const int               height=200,               // yükseklik
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // tutturma
                       const int               scale=2,                  // ölçek
                       const bool              date_scale=true,          // zaman ölç
                       const bool              price_scale=true,         // fiyat ölç
                       const color             clr=clrRed,               // kenar çiz
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,        // kenar çiz
                       const int               point_width=1,            // taşıma no
                       const bool              back=false,               // arkaplanda
                       const bool              selection=false,          // taşıma iç
                       const bool              hidden=true,              // nesne list
                       const long              z_order=0)                // fare tıkla
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- çizelge nesnesini oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_CHART,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": \"çizelge\" nesnesinin oluşturulması başarısız! Hata kodu = ",GetLastEr
      return(false);
     }
//--- nesne koordinatlarının ayarlanması
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- nesne büyüklüğünü ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- tanımlanan nokta koordinatlarına göre çapa (tutturma) köşesini ayarla
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- sembolü ayarla
   ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol);
//--- periyodu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period);
//--- ölçeği ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_CHART_SCALE,scale);
//--- zaman ölçeğini görüntüle (true) veya (false) gizle
   ObjectSetInteger(chart_ID,name,OBJPROP_DATE_SCALE,date_scale);
//--- zaman ölçeğini görüntüle (true) veya (false) gizle
   ObjectSetInteger(chart_ID,name,OBJPROP_PRICE_SCALE,price_scale);
//--- nesne vurgulama modu etkinken kenar çizgisi rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- nesne vurgulama modu etkinken kenar çizgisinin stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- nesne taşıma için tutturma noktası boyutunu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- etiketi fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Sembolü ve Çizelge nesnesinin zaman aralığını ayarla             |
//+------------------------------------------------------------------+
bool ObjectChartSetSymbolAndPeriod(const long            chart_ID=0,        // (nesne
                                   const string          name="Chart",      // nesne is
                                   const string          symbol="EURUSD",   // sembol
                                   const ENUM_TIMEFRAMES period=PERIOD_H1) // zaman a
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- çizelge nesnesinin sembolünü ve zaman aralığını ayarla
   if(!ObjectSetString(chart_ID,name,OBJPROP_SYMBOL,symbol))
     {
      Print(__FUNCTION__,
            ": \"Çizelge\" nesnesinin sembolü ayarlanamadı! Hata kodu = ",GetLastError
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_PERIOD,period))
     {
      Print(__FUNCTION__,
```

```
            ": \"Çizelge\" nesnesinin periyodu ayarlanamadı! Hata kodu = ",GetLastErro
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Çizelge nesnesini taşı                                           |
//+------------------------------------------------------------------+
bool ObjectChartMove(const long   chart_ID=0,   // (nesne olmayan) çizelgenin kimliği
                     const string name="Chart", // nesne ismi
                     const int    x=0,          // X koordinatı
                     const int    y=0)          // Y koordinatı
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesneyi taşı
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": \"Çizelge\" nesnesinin X koordinatı ayarlanamadı! Hata kodu = ",GetLast
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": \"Çizelge\" nesnesinin Y koordinatı ayarlanamadı! Hata kodu = ",GetLast
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Çizelge nesnesinin boyutunu değiştir                             |
//+------------------------------------------------------------------+
bool ObjectChartChangeSize(const long   chart_ID=0,   // (nesne olmayan) çizelgenin ki
                           const string name="Chart", // nesne ismi
                           const int    width=300,    // genişlik
                           const int    height=200)   // yükseklik
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesne boyutunu değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": \"Çizelge\" nesnesinin genişliği ayarlanamadı! Hata kodu = ",GetLastEr
      return(false);
     }
```

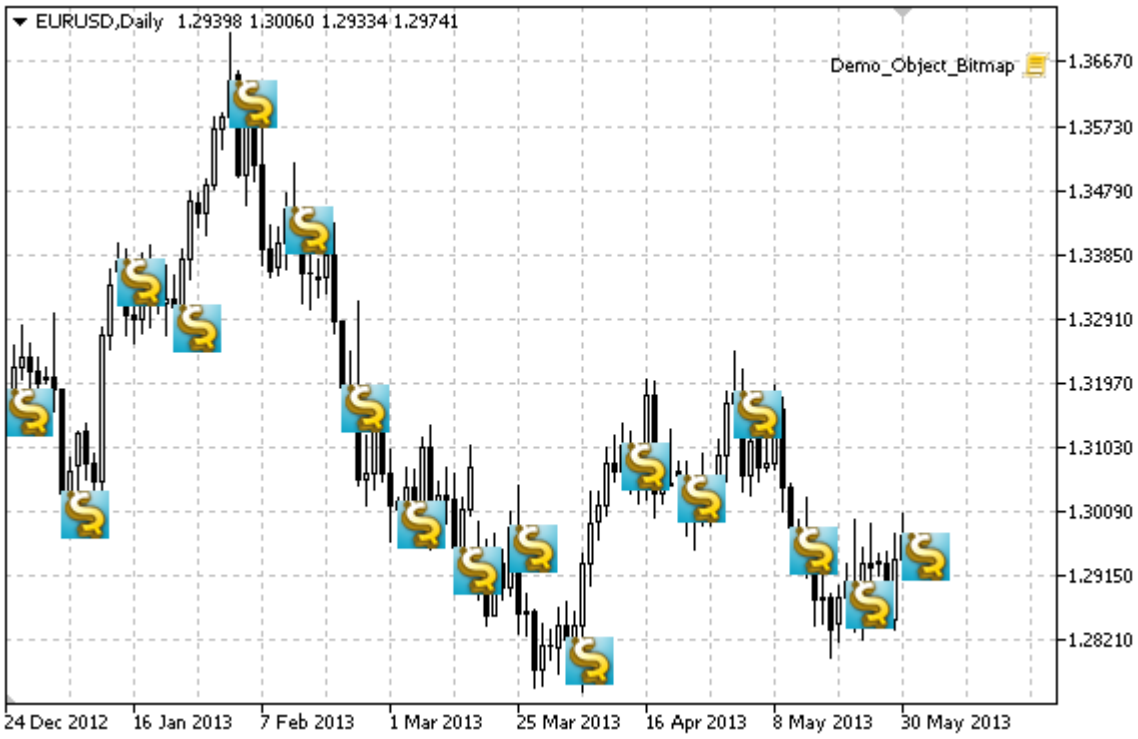
```
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": \"Çizelge\" nesnesinin genişliği ayarlanamadı! Hata kodu = ",GetLastEr
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Çizelge nesnesinin kimliğine dönüş yap                           |
//+------------------------------------------------------------------+
long ObjectChartGetID(const long   chart_ID=0,   // (nesne olmayan) çizelgenin kimliği
                      const string name="Chart") // nesne ismi
  {
//--- Çizelge nesnesinin kimliğini almak için değişkeni hazırla
   long id=-1;
//--- hata değerini sıfırla
   ResetLastError();
//--- kimliği al
   if(!ObjectGetInteger(chart_ID,name,OBJPROP_CHART_ID,0,id))
     {
      Print(__FUNCTION__,
            ": \"Çizelge\" nesnesinin kimliği alınamadı! Hata kodu = ",GetLastError())
     }
//--- sonuç değerine dönüş yap
   return(id);
  }
//+------------------------------------------------------------------+
//| Çizelge nesnesini sil                                            |
//+------------------------------------------------------------------+
bool ObjectChartDelete(const long   chart_ID=0,   // (nesne olmayan) çizelgenin kimliği
                       const string name="Chart") // nesne ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- düğmeyi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Çizelge\" nesnesinin silinmesi başarısız oldu! Hata kodu = ",GetLastE
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

```
void OnStart()
  {
//--- Piyasa Gözlemindeki sembollerin sayısı
   int  symbols=SymbolsTotal(true);
//--- sembol listesinde belirtilen isimde bir sembol var mı kontrol et
   bool exist=false;
   for(int i=0;i<symbols;i++)
      if(InpSymbol==SymbolName(i,true))
        {
         exist=true;
         break;
        }
   if(!exist)
     {
      Print("Hata! ",InpSymbol," sembol, \"Piyasa Gözlemi\" içinde mevcut değil!");
      return;
     }
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpScale<0 || InpScale>5)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }

//--- çizelge penceresi büyüklüğü
   long x_distance;
   long y_distance;
//--- pencere büyüklüğünü ayarla
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Çizelgenin genişlik değeri alınamadı! Hata kodu = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Çizelgenin yükseklik değeri alınamadı! Hata kodu = ",GetLastError());
      return;
     }
//--- Çizelge nesnesinin koordinatlarını ve boyutunu ayarla
   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;
//--- çizelge nesnesini oluştur
   if(!ObjectChartCreate(0,InpName,0,InpSymbol,InpPeriod,x,y,x_size,y_size,InpCorner,I
      InpPriceScale,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,Inp
     {
      return;
     }
```

```
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- Çizelge nesnesini uzat
   int steps=(int)MathMin(x_distance*7/16,y_distance*7/16);
   for(int i=0;i<steps;i++)
     {
      //--- yeniden boyutlandır
      x_size+=1;
      y_size+=1;
      if(!ObjectChartChangeSize(0,InpName,x_size,y_size))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yeniden çiz ve 0.01 saniye bekle
      ChartRedraw();
      Sleep(10);
     }
//--- yarım saniyelik gecikme
   Sleep(500);
//--- çizelge zaman aralığını değiştir
   if(!ObjectChartSetSymbolAndPeriod(0,InpName,InpSymbol,PERIOD_M1))
      return;
   ChartRedraw();
//--- üç saniyelik gecikme
   Sleep(3000);
//--- nesneyi sil
   ObjectChartDelete(0,InpName);
   ChartRedraw();
//--- 1 saniye bekle
   Sleep(1000);
//---
  }
```

OBJ_BITMAP

Bitmap nesnesi.



Not

Bitmap nesnesi için, bir resmin görünürlük alanını seçebilirsin.

Örnek

Aşağıdaki script, çizelge üzerinde çeşitli bitmap'ler oluşturur. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, çizelge penceresi içinde bir bitmap nesnesi oluşturur."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpFile="\\Images\\dollar.bmp"; // Bitmap dosyası ismi
input int             InpWidth=24;                    // Görünürlük alanının X koordin
input int             InpHeight=24;                   // Görünürlük alanının X koordin
input int             InpXOffset=4;                   // Görünürlük alanının X eksenin
input int             InpYOffset=4;                   // Görünürlük alanının Y eksenin
input color           InpColor=clrRed;                // Kenar çizgisi rengi (vurgular
input ENUM_LINE_STYLE InpStyle=STYLE_SOLID;           // Çizgi stili (vurgulandığında)
input int             InpPointWidth=1;                // Taşıma noktası büyüklüğü
input bool            InpBack=false;                  // Arkaplan nesnesi
input bool            InpSelection=false;             // Taşıma için vurgula
```

```
input bool            InpHidden=true;                  // Nesne listesinde gizle
input long            InpZOrder=0;                     // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Çizelge penceresinde bir bitmap nesnesi oluşturur                |
//+------------------------------------------------------------------+
bool BitmapCreate(const long            chart_ID=0,        // çizelge kimliği
                  const string          name="Bitmap",     // bitmap ismi
                  const int             sub_window=0,      // alt-pencere indisi
                  datetime              time=0,            // tutturma noktası zamanı
                  double                price=0,           // çapa noktası fiyatı
                  const string          file="",           // bitmap dosya adı
                  const int             width=10,          // görünürlük alanının X ko
                  const int             height=10,         // görünürlük alanının Y ko
                  const int             x_offset=0,        // görünürlük alanının X el
                  const int             y_offset=0,        // görünürlük alanının Y el
                  const color           clr=clrRed,        // vurgulandığında kenar çi
                  const ENUM_LINE_STYLE style=STYLE_SOLID, // vurgulandığında çizgi re
                  const int             point_width=1,     // taşıma noktasının boyutu
                  const bool            back=false,        // arka-planda
                  const bool            selection=false,   // taşıma için vurgula
                  const bool            hidden=true,       // nesne listesinde gizle
                  const long            z_order=0)         // fare tıklama önceliği
  {
//--- ayarlanmamışsa, tutturma noktası koordinatlarını ayarla
   ChangeBitmapEmptyPoint(time,price);
//--- hata değerini sıfırla
   ResetLastError();
//--- bir bitmap nesnesi oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP,sub_window,time,price))
     {
      Print(__FUNCTION__,
            ": çizelge penceresinde bir bitmap nesnesi oluşturulamadı! Hata kodu = ",G
      return(false);
     }
//--- resim dosyasının adresini ayarla
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            ": resmin yüklenmesi başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- resmin görünürlük alanını ayarla; eğer yükseklik ve genişlik değerleri
//--- sırasıyla kaynak resmin genişlik ve yüksekliğini geçerse
//--- etiket çizilmez; aksi durumda ise,
//--- sadece bu değerler karşılık gelen kısım çizilir
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- resmin görünürlük alanında görüntülenecek kısmını ayarla
//--- öntanımlı kısım resmin sol üst alanıdır; değerler,
```

```
//--- resmin başka bir kısmının gösterilmesi için bir kaydırma gerçekleştirir
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- nesne vurgulama modu etkinken kenar çizgisi rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- nesne vurgulama modu etkinken kenar çizgisinin stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- nesne taşıma için tutturma noktası boyutunu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- etiketi fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Bitmap nesnesi için yeni bir resim ayarla                        |
//+------------------------------------------------------------------+
bool BitmapSetImage(const long   chart_ID=0,    // çizelge kimliği
                    const string name="Bitmap", // bitmap ismi
                    const string file="")       // dosya adresi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- resim dosyasının adresini ayarla
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,file))
     {
      Print(__FUNCTION__,
            ": resmin yüklenmesi başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Bir bitmap nesnesini çizelge üzerinde taşı                       |
//+------------------------------------------------------------------+
bool BitmapMove(const long   chart_ID=0,    // çizelge kimliği
                const string name="Bitmap", // bitmap ismi
                datetime     time=0,        // tutturma noktası zamanı
                double       price=0)       // tutturma noktası fiyatı
  {
//--- noktanın pozisyonu belirlenmemişse, noktayı Bid (alış) fiyatı olan mevcut çubuğ
```

```
   if(!time)
      time=TimeCurrent();
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
//--- hata değerini sıfırla
   ResetLastError();
//--- tutturma noktasını taşı
   if(!ObjectMove(chart_ID,name,0,time,price))
     {
      Print(__FUNCTION__,
            ": tutturma noktasının taşınması başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Nesnenin (bitmap) görünürlük alanı boyutunu değiştir             |
//+------------------------------------------------------------------+
bool BitmapChangeSize(const long   chart_ID=0,     // çizelge kimliği
                      const string name="Bitmap", // bitmap ismi
                      const int    width=0,       // bitmap genişliği
                      const int    height=0)      // bitmap yüksekliği
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- bitmap büyüklüğünü değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": bitmap genişliğinin değiştirilmesi başarısız! Hata kodu = ",GetLastErro
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": bitmap yüksekliğinin değiştirilmesi başarısız! Hata kodu = ",GetLastErr
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Görünürlük alanının üst sağ köşesinin koordinatlarını değiştir   |
//+------------------------------------------------------------------+
bool BitmapMoveVisibleArea(const long   chart_ID=0,     // çizelge tanımlayıcısı
                           const string name="Bitmap", // bitmap ismi
                           const int    x_offset=0,    // görünürlük alanının X koordi
                           const int    y_offset=0)    // görünürlük alanının Y koordi
```

```
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- bitmap nesnesinin görünürlük alanının koordinatlarını değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
      Print(__FUNCTION__,
            ": görünürlük alanının X koordinatının değiştirilmesi başarısız! Hata kodu
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            ": görünürlük alanının X koordinatının değiştirilmesi başarısız! Hata kodu
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Bitmap nesnesini sil                                             |
//+------------------------------------------------------------------+
bool BitmapDelete(const long   chart_ID=0,    // çizelge tanımlayıcısı
                  const string name="Bitmap") // bitmap ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- etiketi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": bitmap nesnesi silinemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Tutturma noktası değerlerini kontrol et ve                       |
//| boş olanlar için varsayılan değerleri ayarla                     |
//+------------------------------------------------------------------+
void ChangeBitmapEmptyPoint(datetime &time,double &price)
  {
//--- noktanın zamanı ayarlanmamışsa, mevcut çubuk üzerinde olacak
   if(!time)
      time=TimeCurrent();
//--- noktanın fiyatı ayarlanmamışsa, Bid değerini alacak
   if(!price)
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
```

```
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date[];  // görünür çubukların tarihlerini depolayan bir dizi
   double   close[]; // kapanış (Close) fiyatlarının depolanması için bir dizi
//--- bitmap dosyasının adı
   string   file="\\Images\\dollar.bmp";
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- bellek tahsisi
   ArrayResize(date,bars);
   ArrayResize(close,bars);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- kapanış (Close) fiyatları dizisini doldur
   if(CopyClose(Symbol(),Period(),0,bars,close)==-1)
     {
      Print("Kapanış fiyatı değerleri kopyalanamadı! Hata kodu = ",GetLastError());
      return;
     }
//--- resmin ne sıklıkta görüntüleneceğini tanımla
   int scale=(int)ChartGetInteger(0,CHART_SCALE);
//--- adım değerini taımla
   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;
```

```
      }
//--- High ve Low (yüksek ve düşük) çubukların değerleri için (boşluklu olarak) bitmap
   for(int i=0;i<bars;i+=step)
     {
      //--- bitmap'leri oluştur
      if(!BitmapCreate(0,"Bitmap_"+(string)i,0,date[i],close[i],InpFile,InpWidth,InpHe
         InpYOffset,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidden,Inp
        {
         return;
        }
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- yarım saniyelik gecikme
   Sleep(500);
//--- satış işaretlerini sil
   for(int i=0;i<bars;i+=step)
     {
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      if(!BitmapDelete(0,"Bitmap_"+(string)i))
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//---
  }
```

OBJ_BITMAP_LABEL

Bitmap Etiketi nesnesi.

Not

İşarete göre tutturma noktası (çapa) konumu ENUM_ARROW_ANCHOR sayımından seçilebilir. Tutturma noktası koordinatları piksel bazında ayarlanır.

Bitmap tutturma köşesini de ayrıca ENUM_BASE_CORNER sayımından seçebilirsiniz.

Bitmap nesnesi için, bir resmin görünürlük alanını seçebilirsiniz.

Örnek

Aşağıdaki script, çizelge üzerinde çeşitli bitmap'ler oluşturur. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script \"Bitmap Etiketi\" nesnesini oluşturur."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string            InpName="BmpLabel";              // Etiket ismi
input string            InpFileOn="\\Images\\dollar.bmp"; // "On" modu için dosya ismi
input string            InpFileOff="\\Images\\euro.bmp";  // "Off" modu için dosya isi
input bool              InpState=false;                   // Etiket basılı/serbest
input ENUM_BASE_CORNER  InpCorner=CORNER_LEFT_UPPER;      // Tutturma (çapa) yapılacal
input ENUM_ANCHOR_POINT InpAnchor=ANCHOR_CENTER;          // Çapa tipi
input color             InpColor=clrRed;                  // Vurgulandığında, kenar çi
```

```
input ENUM_LINE_STYLE   InpStyle=STYLE_SOLID;             // Vurgulandığında çizgi sti
input int               InpPointWidth=1;                  // Taşıma için nokta boyutu
input bool              InpBack=false;                    // Arka-plan nesnesi
input bool              InpSelection=false;               // Taşımak için vurgula
input bool              InpHidden=true;                   // Nesne listesinde gizle
input long              InpZOrder=0;                      // Fare tıklaması için öncel
//+------------------------------------------------------------------+
//| Bitmap nesnesini oluştur                                         |
//+------------------------------------------------------------------+
bool BitmapLabelCreate(const long              chart_ID=0,               // çizelge ki
                       const string            name="BmpLabel",          // etiket isr
                       const int               sub_window=0,             // alt-pencer
                       const int               x=0,                      // X koordina
                       const int               y=0,                      // Y koordina
                       const string            file_on="",               // "On" modur
                       const string            file_off="",              // "Off" modu
                       const int               width=0,                  // X koordina
                       const int               height=0,                 // Y koordina
                       const int               x_offset=10,              // X eksenino
                       const int               y_offset=10,              // Y eksenino
                       const bool              state=false,              // basılı/ser
                       const ENUM_BASE_CORNER  corner=CORNER_LEFT_UPPER, // tutturma
                       const ENUM_ANCHOR_POINT anchor=ANCHOR_LEFT_UPPER, // çapa tipi
                       const color             clr=clrRed,               // kenar çizg
                       const ENUM_LINE_STYLE   style=STYLE_SOLID,        // kenar çizg
                       const int               point_width=1,            // taşıma nok
                       const bool              back=false,               // arkaplanda
                       const bool              selection=false,          // taşıma içi
                       const bool              hidden=true,              // nesne list
                       const long              z_order=0)                // fare tıkla
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- bir bitmap etiketi oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_BITMAP_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": \"Bitmap Etiketi\" nesnesi oluşturulması başarısız! Hata kodu = ",GetLa
      return(false);
     }
//--- On ve Off modları için görüntüleri ayarla
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,0,file_on))
     {
      Print(__FUNCTION__,
            ": "On" modu için resim yüklenemedi! Hata kodu = ",GetLastError());
      return(false);
     }
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,1,file_off))
     {
```

```
      Print(__FUNCTION__,
            ": "Off" modu için resim yüklenemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- etiket koordinatlarını ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- resmin görünürlük alanını ayarla; eğer yükseklik ve genişlik değerleri
//--- sırasıyla kaynak resmin genişlik ve yüksekliğini geçerse
//--- etiket çizilmez; aksi durumda ise,
//--- sadece bu değerler karşılık gelen kısım çizilir
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- resmin görünürlük alanında görüntülenecek kısmını ayarla
//--- öntanımlı kısım resmin sol üst alanıdır; değerler,
//--- resmin başka bir kısmının gösterilmesi için bir kaydırma gerçekleştirir
   ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset);
   ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset);
//--- etiket durumunu tanımla (basılı veya serbest)
   ObjectSetInteger(chart_ID,name,OBJPROP_STATE,state);
//--- tanımlanan nokta koordinatlarına göre çapa (tutturma) köşesini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- tutturma noktası tipini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ANCHOR,anchor);
//--- nesne vurgulama modu etkinken kenar çizgisi rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- nesne vurgulama modu etkinken kenar çizgisinin stilini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- nesne taşıma için tutturma noktası boyutunu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,point_width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- etiketi fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Bitmap etiket nesnesi için yeni bir resim oluştur                |
//+------------------------------------------------------------------+
bool BitmapLabelSetImage(const long   chart_ID=0,      // çizelge kimliği
                         const string name="BmpLabel", // etiket ismi
                         const int    on_off=0,        // şekillendirici (On veya Off)
                         const string file="")         // dosya adresi
```

```
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- resim dosyasının adresini ayarla
   if(!ObjectSetString(chart_ID,name,OBJPROP_BMPFILE,on_off,file))
     {
      Print(__FUNCTION__,
            ": resmin yüklenmesi başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Bitmap Etiket nesnesini taşı                                     |
//+------------------------------------------------------------------+
bool BitmapLabelMove(const long   chart_ID=0,       // çizelge kimliği
                     const string name="BmpLabel", // etiket ismi
                     const int    x=0,             // X koordinatı
                     const int    y=0)             // Y koordinatı
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesneyi taşı
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": nesnenin X koordinatını taşıma başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": nesnenin Y koordinatını taşıma başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Nesnenin görünürlük alanını değiştir                             |
//+------------------------------------------------------------------+
bool BitmapLabelChangeSize(const long   chart_ID=0,       // çizelge kimliği
                           const string name="BmpLabel", // etiket ismi
                           const int    width=0,         // etiket genişliği
                           const int    height=0)        // etiket yüksekliği
  {
//--- hata değerini sıfırla
   ResetLastError();
```

```
//--- nesne boyutunu değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": nesne genişliğinin değiştirilmesi başarısız oldu! Hata kodu = ",GetLast
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": nesne yüksekliğinin değiştirilmesi başarısız oldu! Hata kodu = ",GetLas
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Görünürlük alanının üst sağ köşesinin koordinatlarını değiştir   |
//+------------------------------------------------------------------+
bool BitmapLabelMoveVisibleArea(const long   chart_ID=0,      // çizelge tanımlayıcısı
                                const string name="BmpLabel", // etiket ismi
                                const int    x_offset=0,      // görünürlük alanı X ko
                                const int    y_offset=0)      // görünürlük alanı Y ko
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesnenin görünürlük alanı koordinatlarını değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XOFFSET,x_offset))
     {
      Print(__FUNCTION__,
            ": görünürlük alanının X koordinatının değiştirilmesi başarısız! Hata kodu
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YOFFSET,y_offset))
     {
      Print(__FUNCTION__,
            ": görünürlük alanının X koordinatının değiştirilmesi başarısız! Hata kodu
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| "Bitmap Etiket" nesnesini sil                                    |
//+------------------------------------------------------------------+
bool BitmapLabelDelete(const long   chart_ID=0,      // çizelge kimliği
                       const string name="BmpLabel") // etiket ismi
  {
//--- hata değerini sıfırla
```

```
   ResetLastError();
//--- etiketi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Bitmap etiket\" nesnesinin silinmesi başarısız oldu! Hata kodu = ",Ge
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- çizelge penceresi büyüklüğü
   long x_distance;
   long y_distance;
//--- pencere büyüklüğünü ayarla
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Çizelgenin genişlik değeri alınamadı! Hata kodu = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Çizelgenin yükseklik değeri alınamadı! Hata kodu = ",GetLastError());
      return;
     }
//--- bitmap etiketinin koordinatlarını belirle
   int x=(int)x_distance/2;
   int y=(int)y_distance/2;
//--- etiket büyüklüğünü ve görünürlük alanı koordinatlarını ayarla
   int width=32;
   int height=32;
   int x_offset=0;
   int y_offset=0;
//--- bitmap etiketini pencerenin ortasına yerleştir
   if(!BitmapLabelCreate(0,InpName,0,x,y,InpFileOn,InpFileOff,width,height,x_offset,y_
      InpCorner,InpAnchor,InpColor,InpStyle,InpPointWidth,InpBack,InpSelection,InpHidd
     {
      return;
     }
//--- çizelgeyi yenile ve bir saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- etiketin görünürlük alanı boyutunu, döngü içinde değiştir
   for(int i=0;i<6;i++)
```

```
      {
       //--- görünürlük alanı boyutunu değiştir
       width--;
       height--;
       if(!BitmapLabelChangeSize(0,InpName,width,height))
          return;
       //--- script işlemi devre dışı bırakıldı mı kontrol et
       if(IsStopped())
          return;
       //--- çizelgeyi yenile
       ChartRedraw();
       // 0.3 saniyelik gecikme
       Sleep(300);
      }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- etiketin görünürlük alanı koordinatlarını, döngü içinde değiştir
   for(int i=0;i<2;i++)
     {
       //--- etiketin görünürlük alanı koordinatlarını değiştir
       x_offset++;
       y_offset++;
       if(!BitmapLabelMoveVisibleArea(0,InpName,x_offset,y_offset))
          return;
       //--- script işlemi devre dışı bırakıldı mı kontrol et
       if(IsStopped())
          return;
       //--- çizelgeyi yenile
       ChartRedraw();
       // 0.3 saniyelik gecikme
       Sleep(300);
      }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- etiketi sil
   BitmapLabelDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_EDIT

"Düzenle" nesnesi.

Not

Tutturma noktası koordinatları piksel bazında ayarlanır. "Düzenle" nesnesinin tutturma köşesini ENUM_BASE_CORNER sayımından seçebilirsiniz.

Ayrıca, "Düzenle" nesnesi içinde kullanacağınız metin hizalama tipini de ENUM_ALIGN_MODE sayımından seçebilirsiniz.

Örnek

Aşağıdaki script, "Düzenle" nesnesini oluşturur ve çizelge üzerinde taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Düzenle\" nesnesini çizelge üzerinde oluşturur ."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string            InpName="Edit";              // Nesne ismi
input string            InpText="Text";              // Nesne metni
input string            InpFont="Arial";             // Yazı tipi
input int               InpFontSize=14;              // Yazı tipi büyüklüğü
input ENUM_ALIGN_MODE   InpAlign=ALIGN_CENTER;       // Metin hizalama tipi
input bool              InpReadOnly=false;           // Düzenlenebilirlik
input ENUM_BASE_CORNER  InpCorner=CORNER_LEFT_UPPER; // Tutturma için çizelge köşesi
input color             InpColor=clrBlack;           // Metin rengi
```

```
input color             InpBackColor=clrWhite;       // Arka-plan rengi
input color             InpBorderColor=clrBlack;     // Kenar çizgisi rengi
input bool              InpBack=false;               // Arka-plan nesnesi
input bool              InpSelection=false;          // Taşıma için vurgula
input bool              InpHidden=true;              // Nesne listesinde gizle
input long              InpZOrder=0;                 // Fare tıklaması önceliği
//+------------------------------------------------------------------+
//| "Düzenle" nesnesini oluştur                                      |
//+------------------------------------------------------------------+
bool EditCreate(const long             chart_ID=0,               // çizelge tanımlayıc
                const string           name="Edit",              // nesne ismi
                const int              sub_window=0,             // alt-pencere indisi
                const int              x=0,                      // X koordinatı
                const int              y=0,                      // Y koordinatı
                const int              width=50,                 // genişlik
                const int              height=18,                // yükseklik
                const string           text="Text",              // metin
                const string           font="Arial",             // yazı tipi
                const int              font_size=10,             // yazı tipi boyutu
                const ENUM_ALIGN_MODE  align=ALIGN_CENTER,       // hizalama tipi
                const bool             read_only=false,          // düzenlenebilirlik
                const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // çizelge üzerindeki
                const color            clr=clrBlack,             // metin rengi
                const color            back_clr=clrWhite,        // arka-plan rengi
                const color            border_clr=clrNONE,       // kenar çizgisi renç
                const bool             back=false,               // arka-plan nesnesi
                const bool             selection=false,          // taşıma için vurgul
                const bool             hidden=true,              // nesne listesinde v
                const long             z_order=0)                // Fare tıklaması içi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- düzenleme alanı oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_EDIT,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": \"Düzenle\" nesnesi oluşturulamadı! Hata kodu = ",GetLastError());
      return(false);
     }
//--- nesne koordinatlarının ayarlanması
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- nesne büyüklüğünü ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- metni ayarla
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- metin yazı tipini ayarla
   ObjectSetString(chart_ID,name,OBJPROP_FONT,font);
```

```
//--- yazı tipi boyutunu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_FONTSIZE,font_size);
//--- nesne içindeki metin hizalama tipini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ALIGN,align);
//--- sadece okunur modu etkinleştir (true) veya iptal et (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_READONLY,read_only);
//--- nesnenin tanımlanan koordinatlarına göre, çizelge köşesini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- metin rengini ayara
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- arkaplan rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- kenar çizgisinin rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_COLOR,border_clr);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- etiketi fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| "Düzenle" nesnesini taşı                                         |
//+------------------------------------------------------------------+
bool EditMove(const long   chart_ID=0,  // çizelge kimliği
              const string name="Edit", // nesne ismi
              const int    x=0,         // X koordinatı
              const int    y=0)         // Y koordinatı
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesneyi taşı
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": nesnenin X koordinatını taşıma başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": nesnenin Y koordinatını taşıma başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
```

```
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| "Düzenle" nesnesini yeniden boyutlandır                          |
//+------------------------------------------------------------------+
bool EditChangeSize(const long   chart_ID=0,  // çizelge kimliği
                    const string name="Edit", // nesne ismi
                    const int    width=0,     // genişlik
                    const int    height=0)    // yükseklik
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesne boyutunu değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": nesne genişliğinin değiştirilmesi başarısız oldu! Hata kodu = ",GetLast
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": nesne yüksekliğinin değiştirilmesi başarısız oldu! Hata kodu = ",GetLas
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| "Düzenle" nesnesinin metnini değiştir                            |
//+------------------------------------------------------------------+
bool EditTextChange(const long   chart_ID=0,  // çizelge kimliği
                    const string name="Edit", // nesne ismi
                    const string text="Text") // metin
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesne metnini değiştir
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": metin değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
```

```
//| "Düzenle" nesnesinin metnine dönüş yap                           |
//+------------------------------------------------------------------+
bool EditTextGet(string      &text,         // metin
                 const long   chart_ID=0,  // çizelge tanımlayıcısı
                 const string name="Edit") // nesne ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesne metnini al
   if(!ObjectGetString(chart_ID,name,OBJPROP_TEXT,0,text))
     {
      Print(__FUNCTION__,
            ": metnin alınması başarısız oldu! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| "Düzenle" nesnesini sil                                          |
//+------------------------------------------------------------------+
bool EditDelete(const long   chart_ID=0,  // çizelge kimliği
                const string name="Edit") // nesne ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- etiketi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
            ": \"Düzenle\" nesnesi silinemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- çizelge penceresi büyüklüğü
   long x_distance;
   long y_distance;
//--- pencere büyüklüğünü ayarla
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Çizelgenin genişlik değeri alınamadı! Hata kodu = ",GetLastError());
      return;
```

```
      }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
      {
       Print("Çizelgenin yükseklik değeri alınamadı! Hata kodu = ",GetLastError());
       return;
      }
//--- düzenleme alanını değiştirmek için adım değerini belirle
   int x_step=(int)x_distance/64;
//--- düzenle alanı koordinatlarını ve boyutunu ayarla
   int x=(int)x_distance/8;
   int y=(int)y_distance/2;
   int x_size=(int)x_distance/8;
   int y_size=InpFontSize*2;
//--- metni yerel değişkende sakla
   string text=InpText;
//--- düzenleme alanı oluştur
   if(!EditCreate(0,InpName,0,x,y,x_size,y_size,InpText,InpFont,InpFontSize,InpAlign,
       InpCorner,InpColor,InpBackColor,InpBorderColor,InpBack,InpSelection,InpHidden,In
      {
       return;
      }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- düzenleme alanını uzat
   while(x_size-x<x_distance*5/8)
      {
       //--- düzenleme alanının genişliğini artır
       x_size+=x_step;
       if(!EditChangeSize(0,InpName,x_size,y_size))
          return;
       //--- script işlemi devre dışı bırakıldı mı kontrol et
       if(IsStopped())
          return;
       //--- çizelgeyi yenile ve 0.05 saniye bekle
       ChartRedraw();
       Sleep(50);
      }
//--- yarım saniyelik gecikme
   Sleep(500);
//--- metni değiştir
   for(int i=0;i<20;i++)
      {
       //--- sona ve başa "+" ekle
       text="+"+text+"+";
       if(!EditTextChange(0,InpName,text))
          return;
       //--- script işlemi devre dışı bırakıldı mı kontrol et
       if(IsStopped())
```

```
         return;
      //--- çizelgeyi yeniden çiz ve 0.1 saniye bekle
      ChartRedraw();
      Sleep(100);
     }
//--- yarım saniyelik gecikme
   Sleep(500);
//--- düzenleme alanını sil
   EditDelete(0,InpName);
   ChartRedraw();
//--- 1 saniye bekle
   Sleep(1000);
//---
  }
```

OBJ_EVENT

Olay nesnesi.

Not

Fareyi olayın üzerine getirdiğinizde, içinde yer alan metin gözükecektir.

Örnek

Aşağıdaki script, Olay nesnesini çizelge üzerinde oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Olay\" grafiksel nesnesini çizer."
#property description "Tutturma noktası koordinatları, çizelge penceresi"
#property description "boyutunun yüzdesi şeklinde ayarlanır."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string          InpName="Event";     // Olay ismi
input int             InpDate=25;          // Olay tarihi, %
input string          InpText="Text";      // Olay metni
input color           InpColor=clrRed;     // Olay rengi
input int             InpWidth=1;          // Vurgulandığında nokta büyüklüğü
input bool            InpBack=false;       // Arka-plan olayı
input bool            InpSelection=false; // Taşıma için vurgula
input bool            InpHidden=true;      // Nesne listesinde gizle
```

```
input long            InpZOrder=0;        // Fare tıklaması için öncelik
//+------------------------------------------------------------------+
//| Olay nesnesini çizelge üzerinde oluştur                          |
//+------------------------------------------------------------------+
bool EventCreate(const long            chart_ID=0,       // çizelge kimliği
                 const string          name="Event",     // olay ismi
                 const int             sub_window=0,     // alt-pencere indisi
                 const string          text="Text",      // olay metni
                 datetime              time=0,           // zaman
                 const color           clr=clrRed,       // renk
                 const int             width=1,          // vurgulandığında nokta kalın
                 const bool            back=false,       // arka-planda
                 const bool            selection=false,  // taşıma için vurgula
                 const bool            hidden=true,      // nesne listesinde gizle
                 const long            z_order=0)        // fare tıklaması için öncelik
  {
//--- eğer zaman ayarlanmamışsa, nesneyi son çubuk üzerinde oluştur
   if(!time)
      time=TimeCurrent();
//--- hata değerini sıfırla
   ResetLastError();
//--- Olay nesnesini oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_EVENT,sub_window,time,0))
     {
      Print(__FUNCTION__,
            ": \"Olay\" nesnesini oluşturulması başarısız! Hata kodu = ",GetLastError
      return(false);
     }
//--- olay metnini ayarla
   ObjectSetString(chart_ID,name,OBJPROP_TEXT,text);
//--- rengi ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- nesne vurgulandığında tutturma noktası kalınlığını ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- olayı fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Olay nesnesinin metnini değiştir                                 |
//+------------------------------------------------------------------+
```

```
bool EventTextChange(const long   chart_ID=0,   // çizelge kimliği
                     const string name="Event", // olay ismi
                     const string text="Text")  // metint
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesne metnini değiştir
   if(!ObjectSetString(chart_ID,name,OBJPROP_TEXT,text))
     {
      Print(__FUNCTION__,
            ": metin değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Olay nesnesini taşı                                              |
//+------------------------------------------------------------------+
bool EventMove(const long   chart_ID=0,   // çizelge kimliği
               const string name="Event", // olay ismi
               datetime     time=0)       // zaman
  {
//--- eğer zaman ayarlanmamışsa, olayı son çubuğa ayarla
   if(!time)
      time=TimeCurrent();
//--- hata değerini sıfırla
   ResetLastError();
//--- nesneyi taşı
   if(!ObjectMove(chart_ID,name,0,time,0))
     {
      Print(__FUNCTION__,
            ": \"Olay\" nesnesinin taşınması başarısız oldu! Hata kodu = ",GetLastErro
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Olay nesnesini sil                                               |
//+------------------------------------------------------------------+
bool EventDelete(const long   chart_ID=0,   // çizelge kimliği
                 const string name="Event") // olay ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- nesneyi sil
   if(!ObjectDelete(chart_ID,name))
     {
```

```
      Print(__FUNCTION__,
            ": \"Olay\" nesnesinin silinmesi başarısız oldu! Hata kodu = ",GetLastErro
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- giriş parametrelerinin doğruluğunu kontrol et
   if(InpDate<0 || InpDate>100)
     {
      Print("Hata! Hatalı giriş parametresi değerleri!");
      return;
     }
//--- çizelge penceresindeki görünür çubukların sayısı
   int bars=(int)ChartGetInteger(0,CHART_VISIBLE_BARS);
//--- çizginin tutturma noktası koordinatlarını ayarlamak ve değiştirmek için
//--- tarih değerlerini depolayacak bir dizi
   datetime date[];
//--- bellek tahsisi
   ArrayResize(date,bars);
//--- tarih dizisini doldur
   ResetLastError();
   if(CopyTime(Symbol(),Period(),0,bars,date)==-1)
     {
      Print("Zaman değerlerinin kopyalanması başarısız oldu! Hata kodu = ",GetLastErro
      return;
     }
//--- nesneyi oluşturmak için gereken noktaları tanımla
   int d=InpDate*(bars-1)/100;
//--- Olay nesnesini oluştur
   if(!EventCreate(0,InpName,0,InpText,date[d],InpColor,InpWidth,
      InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- şimdi, nesneyi taşı
//--- döngü sayacı
   int h_steps=bars/2;
//--- nesneyi taşı
   for(int i=0;i<h_steps;i++)
     {
```

```
      //--- bir sonraki değeri kullan
      if(d<bars-1)
         d+=1;
      //--- tutturma noktasını taşı
      if(!EventMove(0,InpName,date[d]))
         return;
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yenile
      ChartRedraw();
      // 0.05 saniyelik gecikme
      Sleep(50);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- kanalı çizelgeden sil
   EventDelete(0,InpName);
   ChartRedraw();
//--- 1 saniyelik gecikme
   Sleep(1000);
//---
  }
```

OBJ_RECTANGLE_LABEL

Dikdörtgen Etiket nesnesi.

Not

Tutturma noktası koordinatları piksel bazında ayarlanır. Dikdörtgen etiket nesnesinin tutturma köşesinin, ENUM_BASE_CORNER sayımı ile seçilmesi mümkündür. Dikdörtgen etiketin kenar çizgileri ise, ENUM_BORDER_TYPE seçile bilir.

Bu nesne, özel grafiksel arayüzler oluşturmak ve tasarlamak için kullanılır.

Örnek

Aşağıdaki script, Dikdörtgen Etiket nesnesini çizelge üzerinde oluşturur ve taşır. Grafiksel nesne özelliklerini oluşturmak ve değiştirmek için özel fonksiyonlar geliştirilmiştir. Bu fonksiyonları kendi uygulamalarınızda da "aynı şekilde" kullanabilirsiniz.

```
//--- açıklama
#property description "Script, \"Dikdörtgen etiket\" grafiksel nesnesini oluşturur."
//--- betiğin çalıştırılması sırasında giriş parametreleri penceresini göster
#property script_show_inputs
//--- betiğin giriş parametreleri
input string           InpName="RectLabel";          // Etiket ismi
input color            InpBackColor=clrSkyBlue;      // Arkaplan rengi
input ENUM_BORDER_TYPE InpBorder=BORDER_FLAT;        // Kenar çizgisi tipi
input ENUM_BASE_CORNER InpCorner=CORNER_LEFT_UPPER; // Tutturma için çizelge köşesi
input color            InpColor=clrDarkBlue;         // Düz kenar çizgisi rengi (Flat)
input ENUM_LINE_STYLE  InpStyle=STYLE_SOLID;         // Düz kenar çizgisi stili (Flat)
input int              InpLineWidth=3;               // Düz kenar çizgisi genişliği (F
```

```
input bool              InpBack=false;               // Arka-plan nesnesi
input bool              InpSelection=true;           // Taşıma için vurgula
input bool              InpHidden=true;              // Nesne listesinde gizle
input long              InpZOrder=0;                 // Fare tıklaması önceliği
//+------------------------------------------------------------------+
//| Dikdörtgen etiketi oluştur                                       |
//+------------------------------------------------------------------+
bool RectLabelCreate(const long             chart_ID=0,               // çizelge tanır
                     const string           name="RectLabel",         // etiket ismi
                     const int              sub_window=0,             // alt pencere i
                     const int              x=0,                      // X koordinatı
                     const int              y=0,                      // Y koordinatı
                     const int              width=50,                 // genişlik
                     const int              height=18,                // yükseklik
                     const color            back_clr=C'236,233,216',  // arkaplan renç
                     const ENUM_BORDER_TYPE border=BORDER_SUNKEN,     // kenar tipi
                     const ENUM_BASE_CORNER corner=CORNER_LEFT_UPPER, // tutturma için
                     const color            clr=clrRed,               // düz kenar ren
                     const ENUM_LINE_STYLE  style=STYLE_SOLID,        // düz kenar sti
                     const int              line_width=1,             // düz kenar kal
                     const bool             back=false,               // arkaplan nesn
                     const bool             selection=false,          // taşıma için v
                     const bool             hidden=true,              // nesne listesi
                     const long             z_order=0)                // fare tıklamas
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- bir dikdörtgen etiket oluştur
   if(!ObjectCreate(chart_ID,name,OBJ_RECTANGLE_LABEL,sub_window,0,0))
     {
      Print(__FUNCTION__,
            ": dikdörtgen etiketin oluşturulması başarısız oldu! Hata kodu = ",GetLast
      return(false);
     }
//--- etiket koordinatlarını ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x);
   ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y);
//--- etiket boyutunu ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width);
   ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height);
//--- arkaplan rengini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_BGCOLOR,back_clr);
//--- kenar çizgisi tipini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border);
//--- tanımlanan nokta koordinatlarına göre çapa (tutturma) köşesini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_CORNER,corner);
//--- düz kenar rengini ayarla (Flat modunda)
   ObjectSetInteger(chart_ID,name,OBJPROP_COLOR,clr);
//--- düz kenar çizgisi stilini ayarla
```

```
   ObjectSetInteger(chart_ID,name,OBJPROP_STYLE,style);
//--- düz kenar çizgisi genişliğini ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_WIDTH,line_width);
//--- ön-planda (false) veya arkaplanda (true) göster
   ObjectSetInteger(chart_ID,name,OBJPROP_BACK,back);
//--- etiketi fare ile taşıma modunu etkinleştir (true) veya devre dışı bırak (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTABLE,selection);
   ObjectSetInteger(chart_ID,name,OBJPROP_SELECTED,selection);
//--- nesne listesinde grafiksel nesnenin adını sakla (true) veya göster (false)
   ObjectSetInteger(chart_ID,name,OBJPROP_HIDDEN,hidden);
//--- çizelge üzerinde fare tıklaması olayının alınması için özellik ayarla
   ObjectSetInteger(chart_ID,name,OBJPROP_ZORDER,z_order);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Dikdörtgen etiketi taşı                                          |
//+------------------------------------------------------------------+
bool RectLabelMove(const long   chart_ID=0,       // çizelge kimliği
                   const string name="RectLabel", // etiket ismi
                   const int    x=0,              // X koordinatı
                   const int    y=0)              // Y koordinatı
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- dikdörtgen etiketi taşı
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XDISTANCE,x))
     {
      Print(__FUNCTION__,
            ": etiketin X koordinatı taşınamadı! Hata kodu = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YDISTANCE,y))
     {
      Print(__FUNCTION__,
            ": etiketin Y koordinatı taşınamadı! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Dikdörtgen etiketin boyutunu değiştir                            |
//+------------------------------------------------------------------+
bool RectLabelChangeSize(const long   chart_ID=0,       // çizelge kimliği
                         const string name="RectLabel", // etiket adı
                         const int    width=50,         // etiket genişliği
                         const int    height=18)        // etiket yüksekliği
  {
```

```
//--- hata değerini sıfırla
   ResetLastError();
//--- etiket boyutunu değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_XSIZE,width))
     {
      Print(__FUNCTION__,
            ": etiketin genişliği değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_YSIZE,height))
     {
      Print(__FUNCTION__,
            ": etiketin yüksekliği değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Dikdörtgen etiketin kenar tipini değiştir                        |
//+------------------------------------------------------------------+
bool RectLabelChangeBorderType(const long             chart_ID=0,           // çizelge
                               const string           name="RectLabel",     // etiket
                               const ENUM_BORDER_TYPE border=BORDER_SUNKEN) // kenar t
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- kenar tipini değiştir
   if(!ObjectSetInteger(chart_ID,name,OBJPROP_BORDER_TYPE,border))
     {
      Print(__FUNCTION__,
            ": kenar tipi değiştirilemedi! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Dikdörtgen etiketi sil                                           |
//+------------------------------------------------------------------+
bool RectLabelDelete(const long   chart_ID=0,       // çizelge kimliği
                     const string name="RectLabel") // etiket ismi
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- etiketi sil
   if(!ObjectDelete(chart_ID,name))
     {
      Print(__FUNCTION__,
```

```
            ": dikdörtgen etiketin silinmesi başarısız! Hata kodu = ",GetLastError());
      return(false);
     }
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- çizelge penceresi büyüklüğü
   long x_distance;
   long y_distance;
//--- pencere büyüklüğünü ayarla
   if(!ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0,x_distance))
     {
      Print("Çizelgenin genişlik değeri alınamadı! Hata kodu = ",GetLastError());
      return;
     }
   if(!ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0,y_distance))
     {
      Print("Çizelgenin yükseklik değeri alınamadı! Hata kodu = ",GetLastError());
      return;
     }
//--- dikdörtgen etiketin koordinatlarını tanımla
   int x=(int)x_distance/4;
   int y=(int)y_distance/4;
//--- etiket boyutunu ayarla
   int width=(int)x_distance/4;
   int height=(int)y_distance/4;
//--- bir dikdörtgen etiket oluştur
   if(!RectLabelCreate(0,InpName,0,x,y,width,height,InpBackColor,InpBorder,InpCorner,
      InpColor,InpStyle,InpLineWidth,InpBack,InpSelection,InpHidden,InpZOrder))
     {
      return;
     }
//--- çizelgeyi yenile ve bir saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- dikdörtgen etiketin boyutunu değiştir
   int steps=(int)MathMin(x_distance/4,y_distance/4);
   for(int i=0;i<steps;i++)
     {
      //--- yeniden boyutlandır
      width+=1;
      height+=1;
      if(!RectLabelChangeSize(0,InpName,width,height))
         return;
```

```
      //--- script işlemi devre dışı bırakıldı mı kontrol et
      if(IsStopped())
         return;
      //--- çizelgeyi yeniden çiz ve 0.01 saniye bekle
      ChartRedraw();
      Sleep(10);
     }
//--- 1 saniyelik gecikme
   Sleep(1000);
//--- kenar tipini değiştir
   if(!RectLabelChangeBorderType(0,InpName,BORDER_RAISED))
      return;
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- kenar tipini değiştir
   if(!RectLabelChangeBorderType(0,InpName,BORDER_SUNKEN))
      return;
//--- çizelgeyi yenile ve 1 saniye bekle
   ChartRedraw();
   Sleep(1000);
//--- etiketi sil
   RectLabelDelete(0,InpName);
   ChartRedraw();
//--- 1 saniye bekle
   Sleep(1000);
//---
  }
```

Nesne Özellikleri

Grafiksel nesneler tiplerine göre çeşitli özelliklere sahip olabilirler. Nesne özellikleri grafiksel nesnelerle çalışmak için kullanılan fonksiyonlar ile ayarlanır veya alınır.

Teknik analizde kullanılan nesnelerinin tümü zaman ve fiyat koordinatlarıyla tutturulur: trend-çizgisi, kanallar, Fibonacci araçları, vb. Ama, kullanıcı deneyimini geliştirmek için sadece çizelgenin görünür kısmına (ana çizelge penceresi veya gösterge pencereleri) bağlanan bazı yardımcı nesneler de mevcuttur.

Nesne	Tanıtıcı	X/Y	Width/ Height	Date/Price	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Text	OBJ_TEXT	–	–	Evet	–	Evet	Evet
Etiket	OBJ_LABEL	Evet	Evet (salt okunur)	–	Evet	Evet	Evet
Düğme	OBJ_BUTTON	Evet	Evet	–	Evet	–	–
Bitmap	OBJ_BITMAP	–	Evet (salt okunur)	Evet	–	Evet	–
Bitmap Etiketi	OBJ_BITMAP_LABEL	Evet	Evet (salt okunur)	–	Evet	Evet	–
Düzenle	OBJ_EDIT	Evet	Evet	–	Evet	–	–
Dikdörtgen Etiket	OBJ_RECTANGLE_LABEL	Evet	Evet	–	Evet	–	–

Tabloda şu tasarımlar kullanılabilir:

- **X/Y** - tutturma noktalarının koordinatları çizelge köşesine göre, pikseller cinsinden belirlenir;
- **Width/Height** - nesnelerin bir yüksekliği ve genişliği vardır. "Salt okunur" modunda genişlik ve yükseklik değerleri sadece bir defa - nesnenin çizelgeye eklenmesi sırasında - hesaplanır;
- **Date/Price** - tutturma noktaları tarih ve fiyat verilerine göre belirlenir;
- **OBJPROP_CORNER** - belirtilen tutturma noktası koordinatlarına göre çizelge köşesini tanımlar. ENUM_BASE_CORNER sayımının 4 değerinden biri olabilir;
- **OBJPROP_ANCHOR** - tutturma noktasını nesne içinde tanımlar ve ENUM_ANCHOR_POINT sayımının 9 değerinden birini alabilir. Koordinatlar bu noktadan seçili çizelge köşesine kadar piksel cinsinden tanımlanır;
- **OBJPROP_ANGLE** - nesne rotasyon açısını sat-yönünün tersine doğru tanımlar.

Nesnelerin oluşturulması ve çizelge üzerinde taşınması için tasarlanmış ObjectCreate() ve ObjectMove() işlevleri gibi, grafiksel nesnelerin özelliklerini tanımlayan fonksiyonlar da, çizelgeye

komutlar göndermek için kullanılır. Bu fonksiyonlar başarılı şekilde çalıştırılırsa, söz konusu değişim komutu çizelge olaylarının genel kuyruğuna eklenir. Çizelge olayları kuyruğu işlendiğinde, grafiksel nesne özelliklerindeki görsel değişimler çizelgeye uygulanır.

Bu sebepten ötürü, bu fonksiyonların çağrılmalarının hemen ardından grafiksel nesneler üzerinde bir güncelleme beklememeniz gerekir. Grafiksel nesneler, genellikle değişim olaylarını takiben, terminal tarafından otomatik olarak güncellenir - yeni bir fiyat teklifinin gelmesinden sonra, çizelge penceresinin yeniden boyutlandırılmasının ardından, vb. Grafiksel nesneleri zorla güncellemek için ChartRedraw() fonksiyonunu kullanın.

ObjectSetInteger() ve ObjectGetInteger() fonksiyonları için

ENUM_OBJECT_PROPERTY_INTEGER

Tanımlayıcı	Açıklama	Özellik Tipi
OBJPROP_COLOR	Renk	color
OBJPROP_STYLE	Stil	ENUM_LINE_STYLE
OBJPROP_WIDTH	Çizgi kalınlığı	int
OBJPROP_BACK	Arkaplandaki nesne	bool
OBJPROP_ZORDER	Çizelge üzerine tıklama olayının (CHARTEVENT_CLICK) için, bir grafikse	long

Tanımlayıcı	Açıklama	Özellik Tipi
	lnesnenin önceliği. Bu değer, nesne oluşturulurken ön tanımlı olarak sıfıra ayarlanır; gerektiğinde öncelik değiştirilebilir. Birnesneyi diğerinin üzerine uygularken, CHARTE	

Tanımlayıcı	Açıklama	Özellik Tipi
	VENT_CLICK olayını, bunların sadece en yüksek öncelikli olanı alacaktır.	
OBJPROP_FILL	Nesneyi renkle doldur (OBJ_RECTANGLE, OBJ_TRIANGLE, OBJ_ELLIPSE, OBJ_CHANNEL, OBJ_STDDE	bool

Tanımlayıcı	Açıklama	Özellik Tipi
	VCHANNEL ve OBJ_REGRESSION nesneleri için)	
OBJPROP_HIDDEN	Nesne listesi için deki bir grafiksel nesnenin ismi nin, terminalin "Grafikler" - "Nesneler" - "nesne listesi" menüsünde göstergi mini	bool

Tanımlayıcı	Açıklama	Özellik Tipi
	yasaklar. true değerine sahip olan nesnenin listede gizlenmesini sağlar. Varsayılan olarak, takvim olayları, alım-satım geçmişi olayları ve MQL5 programları ile oluşturulmuş olaylar	

Tanımlayıcı	Açıklama	Özellik Tipi
	için 'true' değeri ayarlanmıştır. Bunlar gibi grafiksel nesneleri görüntülemek ve özelliklerine erişebilmek için, "nesne listesi" penceresindeki "Tümü" düğmesine basın.	
OBJPROP_SELECTED	Nesle	bool

Tanımlayıcı	Açıklama	Özellik Tipi
	seçildi	
OBJPROP_READONLY	Editnesnesindeki metnin düzenlenmeolanağı	bool
OBJPROP_TYPE	Nesne tipi	ENUM_OBJECT r/o
OBJPROP_TIME	Zaman koordinatı	datetime şekillendirici=tutturma noktası sayısı
OBJPROP_SELECTABLE	Nesnenin mevcudiyeti	bool
OBJPROP_CREATETIME	Nesnenin oluşturulmazamanı	datetime r/o
OBJPROP_LEVELS	Seviyelerinsayısı	int
OBJPROP_LEVELCOLOR	Seviye	color şekillendirici=seviye numarası

Tanımlayıcı	Açıklama	Özellik Tipi
	çizgisinin rengi	
OBJPROP_LEVELSTYLE	Seviye çizgisinin stili stili	ENUM_LINE_STYLE şekillendirici=seviye numarası
OBJPROP_LEVELWIDTH	Seviye çizgisininkalınlığı	int şekillendirici=seviye numarası
OBJPROP_ALIGN	Editnesnesinde (OBJ_EDIT) yatay metin hizalama	ENUM_ALIGN_MODE
OBJPROP_FONTSIZE	Yazı tipi boyutu	int
OBJPROP_RAY_LEFT	Işınsolagider	bool
OBJPROP_RAY_RIGHT	Işınsağa	bool

Tanımlayıcı	Açıklama	Özellik Tipi
	gider	
OBJPROP_RAY	Tüm çizelge pencerelerinin için den geçen dikey çizgi	bool
OBJPROP_ELLIPSE	Fibonacci Yaynes nesi (OBJ_FIBOARC) için bütün bir elipsi gösterir	bool
OBJPROP_ARROWCODE	Oknes nesi için, ok kodu	uchar
OBJPROP_TIMEFRAMES	Birnes neni	bayraklar kümesi flags

Tanımlayıcı	Açıklama	Özellik Tipi
	n farklı zaman dilimlerindeki görünürlüğü	
OBJPROP_ANCHOR	Bir grafiksel nesnenin tutturma noktası konumu	ENUM_ARROW_ANCHOR (OBJ_ARROW için), ENUM_ANCHOR_POINT (OBJ_LABEL, OBJ_BITMAP_LABEL ve OBJ_TEXT için)
OBJPROP_XDISTANCE	Bağlama köşesinden itibaren, X ekseni üzerinde piksel bazında uzaklık (see	int

Tanımlayıcı	Açıklama	Özellik Tipi
	note)	
OBJPROP_YDISTANCE	Bağlama köşesinden itibaren, Y ekseni üzerinde piksel bazında uzaklık (see note)	int
OBJPROP_DIRECTION	Gann nesnesinin trendi	ENUM_GANN_DIRECTION
OBJPROP_DEGREE	Elliott Dalga İşaretinin seviyesi	ENUM_ELLIOT_WAVE_DEGREE
OBJPROP_DRAWLINES	Elliott Dalgasının işar	bool

Tanımlayıcı	Açıklama	Özellik Tipi
	etlenecek eğri çizgileri görüntüleme	
OBJPROP_STATE	Düğme durumu (basılı / serbest)	bool
OBJPROP_CHART_ID	"Çizelge" nesnesinin (OBJ_CHART) tanımlayıcısı. Bu, çizelge işlemlerinde tarif edilen fonksiyonlar kullanılarak,	long r/o

Tanımlayıcı	Açıklama	Özellik Tipi
	nesnenin özellikleri ile normal bir çizelgeymiş gibi çalışılmasını sağlar ama bu durumda bazı istisnalar vardır.	
OBJPROP_XSIZE	X ekseni üzerinde, piksel bazında nesnenin genişliği. OBJ_LABEL (rea	int

Tanımlayıcı	Açıklama	Özellik Tipi
	d only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT ve OBJ_RECTANGLE_LABEL nesneleri için belirlenmiştir.	
OBJPROP_YSIZE	Y ekseni üzerinde, piksel bazı	int

Tanımlayıcı	Açıklama	Özellik Tipi
	ndanesneneyükseklği. OBJ_LABEL (read only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT ve OBJ_RECTANGLE_LABEL nesneleri için belirlen	

Tanımlayıcı	Açıklama	Özellik Tipi
	mişt ir.	
OBJPROP_XOFFSET	"Bit map Lab el" ve "Bit map " (O BJ_ BIT MAP _LA BEL ve OBJ _BIT MAP) graf ikse l nes nele rind eki görü nür dikd örtg en alan ının sol üst köş esin in X koor dina tları . Bu değ er, orji nal	int

Tanımlayıcı	Açıklama	Özellik Tipi
	görüntünün üst sol köşesine göre piksel bazında ayarlanır.	
OBJPROP_YOFFSET	"Bitmap Label" ve "Bitmap" (OBJ_BITMAP_LABEL ve OBJ_BITMAP) grafiksel nesnelerindeki görünür dikdörtgen alan	int

Tanımlayıcı	Açıklama	Özellik Tipi
	ının sol üst köşesinin Y koordinatları. Bu değer, orjinal görüntünün üst sol köşesine göre piksel bazında ayarlanır.	
OBJPROP_PERIOD	"Çizelge" nesnesi için zaman aralığı	ENUM_TIMEFRAMES
OBJPROP_DATE_SCALE	Çizelge nesnesi için zam	bool

Tanımlayıcı	Açıklama	Özellik Tipi
	an ölçeğinin görüntülenmesi	
OBJPROP_PRICE_SCALE	Çizelgenesnesi için fiyat ölçeğinin görüntülenmesi	bool
OBJPROP_CHART_SCALE	Çizelgenesnesi için ölçek	int 0-5 arası değer
OBJPROP_BGCOLOR	OBJ_EDIT, OBJ_BUTTON ve OBJ_RECTANGLE_LABEL için arka	color

Tanımlayıcı	Açıklama	Özellik Tipi
	plan rengi	
OBJPROP_CORNER	Grafiksel nesnenin bağlanabileceği çizelge köşesi	ENUM_BASE_CORNER
OBJPROP_BORDER_TYPE	"Dikdörtgen etiket" nesnesi için kenar tipi	ENUM_BORDER_TYPE
OBJPROP_BORDER_COLOR	OBJ_EDIT ve OBJ_BUTTON nesneleri için kenar çizgisi rengi	color

Çizelge işlemlerinin "Çizelge" nesnesi (OBJ_CHART) için kullanılması durumunda, şu kısıtlamalar uygulanır:

- ChartClose() ile kapatılamaz;
- Sembol/periyot değerleri ChartSetSymbolPeriod() fonksiyonu kullanılarak değiştirilemez;
- Şu özellikler etkisizdir: CHART_SCALE, CHART_BRING_TO_TOP, CHART_SHOW_DATE_SCALE ve CHART_SHOW_PRICE_SCALE (ENUM_CHART_PROPERTY_INTEGER).

OBJ_BITMAP_LABEL ve OBJ_BITMAP, nesneleri için özel bir resim görüntüleme modu, programlanabilir olarak ayarlanabilir. Bu modda, orjinal resmin sadece bir kısmı (görünür dikdörtgen alanın uygulandığı kısım) görüntülenir, resmin diğer kısmı ise görünmez olur. Bu alanın boyutları, OBJPROP_XSIZE ve OBJPROP_YSIZE özellikleri ile ayarlanmalıdır. Görünür alan, OBJPROP_XOFFSET ve OBJPROP_YOFFSET özelliklerini kullanarak, sadece orjinal resim içinde "taşınabilir".

Sabit boyutlu nesneler için (OBJ_BUTTON, OBJ_RECTANGLE_LABEL, OBJ_EDIT ve OBJ_CHART), OBJPROP_XDISTANCE ve OBJPROP_YDISTANCE özellikleri nesnenin sol üst köşesini çizelge köşesine göre (OBJPROP_CORNER) ayarlar, burada X ve Y koordinatları pikseller cinsinden belirlenir.

ObjectSetDouble() ve ObjectGetDouble() fonksiyonları için

ENUM_OBJECT_PROPERTY_DOUBLE

Tanımlayıcı	Açıklama	Özellik Tipi
OBJPROP_PRICE	Fiyat koordinatı	double şekillendirici=tutturma noktası sayısı
OBJPROP_LEVELVALUE	Seviye değeri	double şekillendirici = seviye numarası
OBJPROP_SCALE	Ölçek (Gann nesnelerinin ve Fibonacc	double

Tanımlayıcı	Açıklama	Özellik Tipi
	i Yayların özellikleri)	
OBJPROP_ANGLE	Açı. Bir programtarafından açı belirtilmeden oluşturulmuş nesneler için, boş değere eşittir EMPTY_VALUE	double
OBJPROP_DEVIATION	Standart Sapma Kanalı için sapma	double

Tanımlayıcı	Açıklama	Özellik Tipi
	değeri	

ObjectSetString() ve ObjectGetString() fonksiyonları için

ENUM_OBJECT_PROPERTY_STRING

Tanımlayıcı	Açıklama	Özellik Tipi
OBJPROP_NAME	Nesne ismi	string
OBJPROP_TEXT	Nesne açıklaması (nesne için deki metin)	string
OBJPROP_TOOLTIP	Araç ipucu metni. Eğer özellik ayarlanmamışsa, o zaman araç ipucu met	string

Tanımlayıcı	Açıklama	Özellik Tipi
	ni, terminal tarafından otomatik olarak ayarlanır. Bir araç ipucu "\n" (satır başı karakteri) değerinin ayarlanmasıyla devre dışı bırakılabilir	
OBJPROP_LEVELTEXT	Seviye açıklaması	string şekillendirici=seviye numarası
OBJPROP_FONT	Yazı tipi	string

Tanımlayıcı	Açıklama	Özellik Tipi
OBJPROP_BMPFILE	Bitmap etiketi için BMP-dosyasının ismi. Bakınız Kaynaklar	string işleyici: 0-durumu ON, 1-durumu OFF
OBJPROP_SYMBOL	Çizelge nesnesinin sembolü	string

OBJ_RECTANGLE_LABEL nesnesi ("Dikdörtgen etiket") için, aşağıda belirtilen ve ENUM_BORDER_TYPE sayımına karşılık gelen iki tasarım modundan biri ayarlanabilir.

ENUM_BORDER_TYPE

Tanımlayıcı	Açıklama
BORDER_FLAT	Düz biçim
BORDER_RAISED	Çıkıntılı biçim
BORDER_SUNKEN	Konkav biçim

OBJ_EDIT nesnesi ("Düzenle") ve ChartScreenShot() fonksiyonu için, ENUM_ALIGN_MODE sayımının değerlerini kullanarak yatay hizalamayı ayarlayabilirsiniz.

ENUM_ALIGN_MODE

Tanımlayıcı	Açıklama
ALIGN_LEFT	Sola hizalama
ALIGN_CENTER	Ortalanmış (sadece Edit nesnesi için)
ALIGN_RIGHT	Sağa hizalama

Örnek:

```
#define  UP          "\x0431"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   string label_name="my_OBJ_LABEL_object";
   if(ObjectFind(0,label_name)<0)
     {
      Print(,label_name," nesnesi bulunamadı. Hata kodu = ",GetLastError());
      //--- Label nesnesini oluştur
      ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
      //--- X koordinatını ayarla
      ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
      //--- Y koordinatını ayarla
      ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
      //--- metin rengini tanımla
      ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrWhite);
      //--- Label nesnesi için metin tanımla
      ObjectSetString(0,label_name,OBJPROP_TEXT,UP);
      //--- yazı tipini tanımla
      ObjectSetString(0,label_name,OBJPROP_FONT,"Wingdings");
      //--- yazı tipi boyutunu tanımla
      ObjectSetInteger(0,label_name,OBJPROP_FONTSIZE,10);
      //--- saat yönünde 45 derecelik rotasyon
      ObjectSetDouble(0,label_name,OBJPROP_ANGLE,-45);
      //--- fare seçimini devre dışı bırak
      ObjectSetInteger(0,label_name,OBJPROP_SELECTABLE,false);
      //--- bunu çizelge üzerinde çiz
      ChartRedraw(0);
     }
  }
```

Nesne Tutturma Yöntemleri

Metinler, Etiketler, Biteşlem ve Biteşlem etiketi (OBJ_TEXT, OBJ_LABEL, OBJ_BITMAP ve OBJ_BITMAP_LABEL) grafiksel nesneleri, OBJPROP_ANCHOR özelliği ile tanımanan 9 farklı tutturma yöntemin birini içerebilirler.

Nesne	Tanıtıcı	X/Y	Width/Height	Date/Price	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Text	OBJ_TEXT	–	–	Evet	–	**Evet**	Evet
Etiket	OBJ_LABEL	Evet	Evet (salt okunur)	–	Evet	**Evet**	Evet
Düğme	OBJ_BUTTON	Evet	Evet	–	Evet	–	–
Bitmap	OBJ_BITMAP	–	Evet (salt okunur)	Evet	–	**Evet**	–
Bitmap Etiketi	OBJ_BITMAP_LABEL	Evet	Evet (salt okunur)	–	Evet	**Evet**	–
Düzenle	OBJ_EDIT	Evet	Evet	–	Evet	–	–
Dikdörtgen Etiket	OBJ_RECTANGLE_LABEL	Evet	Evet	–	Evet	–	–

Tabloda şu tasarımlar kullanılabilir:

- **X/Y** - tutturma noktalarının koordinatları çizelge köşesine göre, pikseller cinsinden belirlenir;
- **Width/Height** - nesnelerin bir yüksekliği ve genişliği vardır. "Salt okunur" modunda genişlik ve yükseklik değerleri sadece bir defa - nesnenin çizelgeye eklenmesi sırasında - hesaplanır;
- **Date/Price** - tutturma noktaları tarih ve fiyat verilerine göre belirlenir;
- **OBJPROP_CORNER** - belirtilen tutturma noktası koordinatlarına göre çizelge köşesini tanımlar. ENUM_BASE_CORNER sayımının 4 değerinden biri olabilir;
- **OBJPROP_ANCHOR** - tutturma noktasını nesne içinde tanımlar ve ENUM_ANCHOR_POINT sayımının 9 değerinden birini alabilir. Koordinatlar bu noktadan seçili çizelge köşesine kadar piksel cinsinden tanımlanır;
- **OBJPROP_ANGLE** - nesne rotasyon açısını sat-yönünün tersine doğru tanımlar.

İstenen seçenek, ObjectSetInteger(çizelge_tanıtıcısı, nesne_ismi, OBJPROP_ANCHOR, tutturma_noktası_modu) fonksiyonu ile belirtilebilir . Burada tutturma_noktası_modu, ENUM_ANCHOR_POINT sayımının değerlerinden biridir.

ENUM_ANCHOR_POINT

Tanıtıcı	Açıklama
ANCHOR_LEFT_UPPER	Sol üst köşedeki tutturma noktası koordinatı

Tanıtıcı	Açıklama
ANCHOR_LEFT	Merkezden sola doğru tutturma noktası
ANCHOR_LEFT_LOWER	Sol alt köşedeki tutturma noktası
ANCHOR_LOWER	Merkezin aşağısındaki tutturma noktası
ANCHOR_RIGHT_LOWER	Alt sağ köşedeki tutturma noktası
ANCHOR_RIGHT	Merkezden sağa doğru tutturma noktası
ANCHOR_RIGHT_UPPER	Sağ üst köşedeki tutturma noktası
ANCHOR_UPPER	Merkezin üstündeki tutturma noktası
ANCHOR_CENTER	Nesnenin tam ortasındaki tutturma noktası

OBJ_BUTTON, OBJ_RECTANGLE_LABEL, OBJ_EDIT ve OBJ_CHART nesnelerinin üst sol köşelerinde sabit tutturma noktaları bulunur (ANCHOR_LEFT_UPPER).

Örnek:

```
   string text_name="my_OBJ_TEXT_object";
   if(ObjectFind(0,text_name)<0)
     {
      Print("Nesne ",text_name," bulunamadı. Hata kodu = ",GetLastError());
      //--- Çizelgedeki maksimal fiyat değerini al
      double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
      //--- Label nesnesini oluştur
      ObjectCreate(0,text_name,OBJ_TEXT,0,TimeCurrent(),chart_max_price);
      //--- Metin rengini ayarla
      ObjectSetInteger(0,text_name,OBJPROP_COLOR,clrWhite);
      //--- Arkaplan rengini ayarla
      ObjectSetInteger(0,text_name,OBJPROP_BGCOLOR,clrGreen);
      //--- Label nesnesi için metni ayarla
      ObjectSetString(0,text_name,OBJPROP_TEXT,TimeToString(TimeCurrent()));
      //--- Metin yazı-tipini ayarla
      ObjectSetString(0,text_name,OBJPROP_FONT,"Trebuchet MS");
      //--- Yazı-tipi boyutunu ayarla
      ObjectSetInteger(0,text_name,OBJPROP_FONTSIZE,10);
      //--- Sağ üst köşeye bağla
      ObjectSetInteger(0,text_name,OBJPROP_ANCHOR,ANCHOR_RIGHT_UPPER);
      //--- Saat yönünün tersine 90 derece çevir
      ObjectSetDouble(0,text_name,OBJPROP_ANGLE,90);
      //--- Nesnenin fare ile seçilmesini yasakla
      ObjectSetInteger(0,text_name,OBJPROP_SELECTABLE,false);
      //--- nesneyi yeniden çiz
      ChartRedraw(0);
     }
```

Arrow grafiksel nesnesi (OBJ_ARROW) koordinat bağlama için sadece iki seçeneğe sahiptir. Tanımlayıcılar, ENUM_ARROW_ANCHOR sayımı içinde listelenmiştir.

ENUM_ARROW_ANCHOR

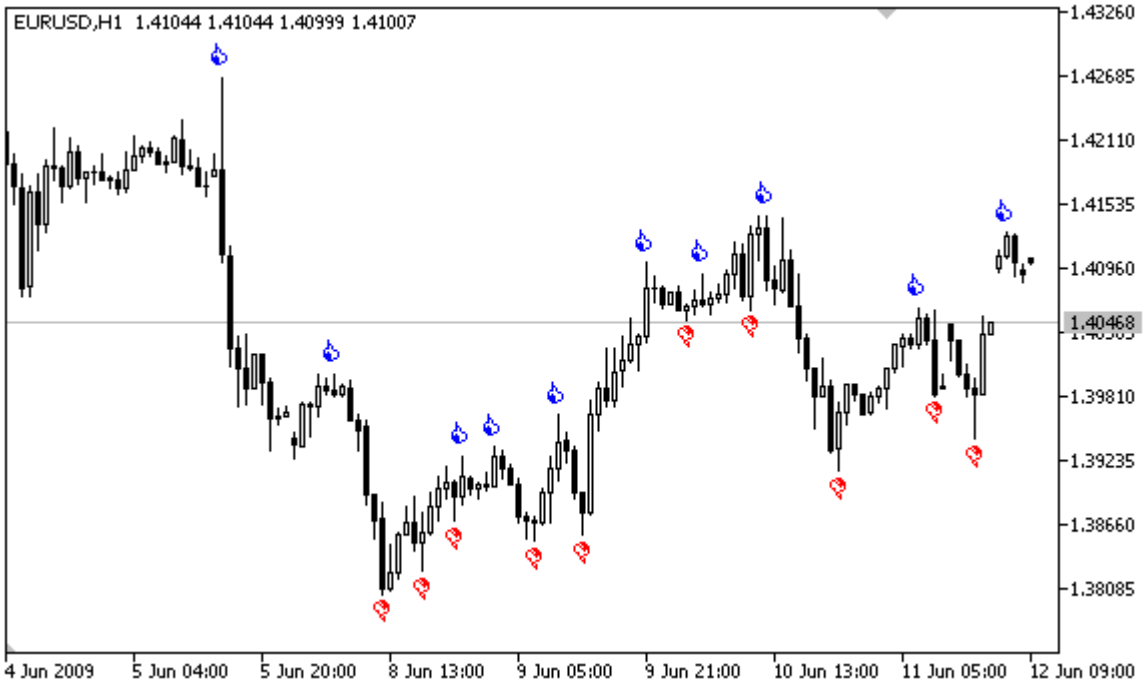
Tanıtıcı	Açıklama
ANCHOR_TOP	Üst kısımdaki ok için tutturma noktası
ANCHOR_BOTTOM	Alt kısımdaki ok için tutturma noktası

Örnek:

```
void OnStart()
  {
//--- Yardımcı diziler
   double Ups[],Downs[];
   datetime Time[];
//--- Dizileri zaman serileri şeklinde ayarla
   ArraySetAsSeries(Ups,true);
   ArraySetAsSeries(Downs,true);
   ArraySetAsSeries(Time,true);
//--- Fractals göstergesinin tanıtıcı değerini oluştur
   int FractalsHandle=iFractals(NULL,0);
   Print("FractalsHandle = ",FractalsHandle);
//--- Son hata değerini sıfır olarak ayarla
   ResetLastError();
//--- Gösterge değerlerini kopyalamayı dene
   int copied=CopyBuffer(FractalsHandle,0,0,1000,Ups);
   if(copied<=0)
     {
      Print("Üsteki fraktaller kopyalanamıyor. Hata = ",GetLastError());
      return;
     }

   ResetLastError();
//--- Gösterge değerlerini kopyalamayı dene
   copied=CopyBuffer(FractalsHandle,1,0,1000,Downs);
   if(copied<=0)
     {
      Print("Alttaki fraktaller kopyalanamıyor. Hata = ",GetLastError());
      return;
     }

   ResetLastError();
//--- Son 1000 çubuğun açılış verisini içeren zaman serisini kopyala
   copied=CopyTime(NULL,0,0,1000,Time);
   if(copied<=0)
     {
      Print("Son 1000 çubuğun açılış zamanını kopyalama işlemi başarısız oldu");
      return;
     }
```

```
   int upcounter=0,downcounter=0; // okların sayısını al
   bool created;// nesne oluşturma denemesinin sonucunu al
   for(int i=2;i<copied;i++)// iFractals göstergesinin değerlerini gözden geçir
     {
      if(Ups[i]!=EMPTY_VALUE)// Üst fraktal bulundu
        {
         if(upcounter<10)// 10 "Yukarı" (Up) oktan fazlasını oluşturma
           {
            //--- Bir "Up" nesnesi oluşturmayı dene
            created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_UP,0,Time[i],Ups[i]
            if(created)// Oluşturulduysa - ayarlamayı yapalım
              {
               //--- Çubuğu örtmemesi için tutturma noktası aşağıda olacak
               ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_BOTTOM);
               //--- Son rötuş - boyayalım
               ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrBlue);
               upcounter++;
              }
           }
        }
      if(Downs[i]!=EMPTY_VALUE)// Daha düşük bir fraktal bulundu
        {
         if(downcounter<10)// 10 "Aşağı" (Down) oktan fazlasını oluşturma
           {
            //--- Bir "Down" nesnesi oluşturmayı dene
            created=ObjectCreate(0,string(Time[i]),OBJ_ARROW_THUMB_DOWN,0,Time[i],Down
            if(created)// Oluşturulduysa - ayarlamayı yapalım
              {
               //--- Çubuğu örtmemesi için tutturma noktası yukarıda olacak
               ObjectSetInteger(0,string(Time[i]),OBJPROP_ANCHOR,ANCHOR_TOP);
               //--- Son rötuş - boyayalım
               ObjectSetInteger(0,string(Time[i]),OBJPROP_COLOR,clrRed);
               downcounter++;
              }
           }
        }
     }
  }
```

Betiğin çalıştırılmasından sonra çizelge bu resimdeki gibi görünecek.



Bir Nesnenin İliştirildiği Çizelge Köşesi

Koordinatlarını çizelgenin bir köşesine göre, piksel cinsinden ayarlayabileceğiniz bir takım grafiksel nesneler vardır. Bu nesneler, tipleriyle birlikte şöyle gösterilebilir (nesne tipi, parantez içinde belirtilir):

- Label (OBJ_LABEL);
- Button (OBJ_BUTTON);
- Bitmap Label (OBJ_BITMAP_LABEL);
- Edit (OBJ_EDIT).
- Rectangle Label (OBJ_RECTANGLE_LABEL);

Nesne	Tanıtıcı	X/Y	Width/Height	Date/Price	OBJPROP_CORNER	OBJPROP_ANCHOR	OBJPROP_ANGLE
Metin	OBJ_TEXT	–	–	Evet	–	Evet	Evet
Etiket	OBJ_LABEL	Evet	Evet (salt okunur)	–	**Evet**	Evet	Evet
Düğme	OBJ_BUTTON	Evet	Evet	–	**Evet**	–	–
Bitmap	OBJ_BITMAP	–	Evet (salt okunur)	Evet	–	Evet	–
Bitmap Etiketi	OBJ_BITMAP_LABEL	Evet	Evet (salt okunur)	–	**Evet**	Evet	–
Düzenle	OBJ_EDIT	Evet	Evet	–	**Evet**	–	–
Dikdörtgen Etiket	OBJ_RECTANGLE_LABEL	Evet	Evet	–	**Evet**	–	–

Tabloda şu tasarımlar kullanılabilir:

- **X/Y** - tutturma noktalarının koordinatları çizelge köşesine göre, pikseller cinsinden belirlenir;
- **Width/Height** - nesnelerin bir yüksekliği ve genişliği vardır. "Salt okunur" modunda genişlik ve yükseklik değerleri sadece bir defa - nesnenin çizelgeye eklenmesi sırasında - hesaplanır;
- **Date/Price** - tutturma noktaları tarih ve fiyat verilerine göre belirlenir;
- **OBJPROP_CORNER** - belirtilen tutturma noktası koordinatlarına göre çizelge köşesini tanımlar. ENUM_BASE_CORNER sayımının 4 değerinden biri olabilir;
- **OBJPROP_ANCHOR** - tutturma noktasını nesne içinde tanımlar ve ENUM_ANCHOR_POINT sayımının 9 değerinden birini alabilir. Koordinatlar bu noktadan seçili çizelge köşesine kadar piksel cinsinden tanımlanır;
- **OBJPROP_ANGLE** - nesne rotasyon açısını sat-yönünün tersine doğru tanımlar.

X ve Y koordinatlarının piksel bazında ölçüleceği çizelge köşesi belirtmek için, ObjectSetInteger(chartID, name, OBJPROP_CORNER, **chart_corner**) fonksiyonunu kullanın, burada:

- chartID - çizelge tanıtıcısıdır;
- name - grafiksel nesnenin ismidir;
- OBJPROP_CORNER - bağlama köşesinin belirlenmesi için özellik tanımlayıcısıdır;
- **chart_corner** - istenen çizelge köşesidir, ENUM_BASE_CORNER sayımının değerlerinden biri olabilir.

ENUM_BASE_CORNER

Tanıtıcı	Açıklama
CORNER_LEFT_UPPER	Çizelgenin sol üst köşesindeki koordinat merkezi
CORNER_LEFT_LOWER	Çizelgenin sol alt köşesindeki koordinat merkezi
CORNER_RIGHT_LOWER	Çizelgenin sağ alt köşesindeki koordinat merkezi
CORNER_RIGHT_UPPER	Çizelgenin sağ üst köşesindeki koordinat merkezi

Örnek:

```
void CreateLabel(long   chart_id,
                 string name,
                 int    chart_corner,
                 string text_label,
                 int    x_ord,
                 int    y_ord)
  {
//---
   ObjectCreate(chart_id,name,OBJ_LABEL,0,0,0);
   ResetLastError();
   if(!ObjectSetInteger(chart_id,name,OBJPROP_CORNER,chart_corner))
      Print("Nesneyi bağlamak için açı ayarlanamadı ",
            name,", hata kodu ",GetLastError());
   ObjectSetInteger(chart_id,name,OBJPROP_XDISTANCE,x_ord);
   ObjectSetInteger(chart_id,name,OBJPROP_YDISTANCE,y_ord);
   ObjectSetString(chart_id,name,OBJPROP_TEXT,text_label);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
   int width=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,"0",50,50);
   CreateLabel(0,arrows[1],CORNER_RIGHT_UPPER,"1",50,50);
   CreateLabel(0,arrows[2],CORNER_RIGHT_LOWER,"2",50,50);
   CreateLabel(0,arrows[3],CORNER_LEFT_LOWER,"3",50,50);
  }
```

Nesnelerin Görünürlükleri

Nesne görünürlük bayraklarının kombinasyonu, nesnenin görünür olduğu çizelge zaman dilimlerini belirler. OBJPROP_TIMEFRAMES özelliğini ayarlamak/almak için, ObjectSetInteger()/ObjectGetInteger() fonksiyonlarını kullanabilirsiniz.

Tanıtıcı	Değer	Açıklama
OBJ_NO_PERIODS	0	Nesne tüm zaman dilimlerinde çizilmez
OBJ_PERIOD_M1	0x00000001	Nesne 1-dakikalık çizelgede çizilir
OBJ_PERIOD_M2	0x00000002	Nesne 2-dakikalık çizelgede çizilir
OBJ_PERIOD_M3	0x00000004	Nesne 3-dakikalık çizelgede çizilir
OBJ_PERIOD_M4	0x00000008	Nesne 4-dakikalık çizelgede çizilir
OBJ_PERIOD_M5	0x00000010	Nesne 5-dakikalık çizelgede çizilir
OBJ_PERIOD_M6	0x00000020	Nesne 6-dakikalık çizelgede çizilir
OBJ_PERIOD_M10	0x00000040	Nesne 10-dakikalık çizelgede çizilir
OBJ_PERIOD_M12	0x00000080	Nesne 12-dakikalık çizelgede çizilir
OBJ_PERIOD_M15	0x00000100	Nesne 15-dakikalık çizelgede çizilir
OBJ_PERIOD_M20	0x00000200	Nesne 20-dakikalık çizelgede çizilir
OBJ_PERIOD_M30	0x00000400	Nesne 30-dakikalık çizelgede çizilir
OBJ_PERIOD_H1	0x00000800	Nesne 1-saatlik çizelgede çizilir
OBJ_PERIOD_H2	0x00001000	Nesne 2-saatlik çizelgede çizilir
OBJ_PERIOD_H3	0x00002000	Nesne 3-saatlik çizelgede çizilir
OBJ_PERIOD_H4	0x00004000	Nesne 4-saatlik çizelgede çizilir

Tanıtıcı	Değer	Açıklama
OBJ_PERIOD_H6	0x00008000	Nesne 6-saatlik çizelgede çizilir
OBJ_PERIOD_H8	0x00010000	Nesne 8-saatlik çizelgede çizilir
OBJ_PERIOD_H12	0x00020000	Nesne 12-saatlik çizelgede çizilir
OBJ_PERIOD_D1	0x00040000	Nesne günlük çizelgelerde çizilir
OBJ_PERIOD_W1	0x00080000	Nesne haftalık çizelgelerde çizilir
OBJ_PERIOD_MN1	0x00100000	Nesne haftalık çizelgelerde çizilir
OBJ_ALL_PERIODS	0x001fffff	Nesne tüm zaman dilimlerinde çizilir

Görünürlük bayrakları "|" sembolü kullanılarak kombine edilebilir. Örneğin, OBJ_PERIOD_M10|OBJ_PERIOD_H4 kombinasyonu, nesnenin 10-dakikalık ve 4-saatlik zaman dilimlerinde görüntüleneceğini belirtir.

Örnek:

```
void OnStart()
  {
//---
   string highlevel="PreviousDayHigh";
   string lowlevel="PreviousDayLow";
   double prevHigh;           // Önceki günün yüksek (High) fiyatı
   double prevLow;            // Önceki günün düşük (Low) fiyatı
   double highs[],lows[];     // High ve Low için diziler

//--- Son hata değerini sıfırla
   ResetLastError();
//--- Günlük zaman aralığındaki son 2 High değerini al
   int highsgot=CopyHigh(Symbol(),PERIOD_D1,0,2,highs);
   if(highsgot>0) // Eğer kopyalama başarılı olduysa
     {
      Print("Son iki günün High (yüksek) fiyatları başarıyla alındı");
      prevHigh=highs[0]; // Önceki günün High değeri
      Print("prevHigh = ",prevHigh);
      if(ObjectFind(0,highlevel)<0) // highlevel isminde bir nesne bulunamadı
        {
         ObjectCreate(0,highlevel,OBJ_HLINE,0,0,0); // Yatal çizgi nesnesini oluştur
        }
      //--- highlevel çizgisi için fiyat seviyesinin değerini ayarla
      ObjectSetDouble(0,highlevel,OBJPROP_PRICE,0,prevHigh);
```

```
      //--- Görünürlüğü sadece PERIOD_M10 ve PERIOD_H4 için ayarla
      ObjectSetInteger(0,highlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
     }
   else
     {
      Print("Son 2 günün High fiyatları alınamadı, Hata = ",GetLastError());
     }

//--- Son hata değerini sıfırla
   ResetLastError();
//--- Son 2 günün, günlük High fiyatlarını al
   int lowsgot=CopyLow(Symbol(),PERIOD_D1,0,2,lows);
   if(lowsgot>0) // Eğer kopyalama başarılıysa
     {
      Print("Son iki günün Low (düşük) fiyatları başarıyla alındı");
      prevLow=lows[0]; // Önceki günün Low değeri
      Print("prevLow = ",prevLow);
      if(ObjectFind(0,lowlevel)<0) // lowlevel isimli bir nesne bulunmadı
        {
         ObjectCreate(0,lowlevel,OBJ_HLINE,0,0,0); // Yatay Çizgi nesnesini oluştur
        }
      //--- lowlevel çizgisi fiyat seviyesini ayarla
      ObjectSetDouble(0,lowlevel,OBJPROP_PRICE,0,prevLow);
      //--- Görünürlüğü sadece PERIOD_M10 ve PERIOD_H4 için ayarla
      ObjectSetInteger(0,lowlevel,OBJPROP_TIMEFRAMES,OBJ_PERIOD_M10|OBJ_PERIOD_H4);
     }
   else Print("Son iki gün için Low (düşük) fiyatlar alınamadı, Error = ",GetLastErro

   ChartRedraw(0); // çizelgeyi zorla yenile
  }
```

Ayrıca Bakınız

PeriodSeconds, Period, Çizelge zaman aralıkları, Zaman ve Tarih

Elliott Dalga Seviyeleri

Elliott Dalgaları, OBJ_ELLIOTWAVE5 ve OBJ_ELLIOTWAVE3 tipi iki grafiksel nesne ile temsil edilir. Dalga boyutunu ayarlamak için (dalga etiketleme yöntemi), OBJPROP_DEGREE özelliği kullanılır, bu özellik için ENUM_ELLIOT_WAVE_DEGREE sayımının değerleri kullanılır.

ENUM_ELLIOT_WAVE_DEGREE

Tanıtıcı	Açıklama
ELLIOTT_GRAND_SUPERCYCLE	Büyük Süper Döngü (Grand Supercycle)
ELLIOTT_SUPERCYCLE	Süper döngü (Supercycle)
ELLIOTT_CYCLE	Döngü (Cycle)
ELLIOTT_PRIMARY	Birincil
ELLIOTT_INTERMEDIATE	Orta (Intermediate)
ELLIOTT_MINOR	İkincil (Minor)
ELLIOTT_MINUTE	Dakika
ELLIOTT_MINUETTE	Saniye (Minuette)
ELLIOTT_SUBMINUETTE	Saniye-altı (Subminuette)

Örnek:

```
   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)))
        {
         //--- INTERMEDIATE'taki imleme seviyesini ayarla
         ObjectSetInteger(0,currobj,OBJPROP_DEGREE,ELLIOTT_INTERMEDIATE);
         //--- dalga tepeleri arasında çizgi göster
         ObjectSetInteger(0,currobj,OBJPROP_DRAWLINES,true);
         //--- çizgi rengini ayarla
         ObjectSetInteger(0,currobj,OBJPROP_COLOR,clrBlue);
         //--- çizgi kalınlığını ayarla
         ObjectSetInteger(0,currobj,OBJPROP_WIDTH,5);
         //--- açıklamayı ayarla
         ObjectSetString(0,currobj,OBJPROP_TEXT,"test script");
        }
     }
```

Gann Nesneleri

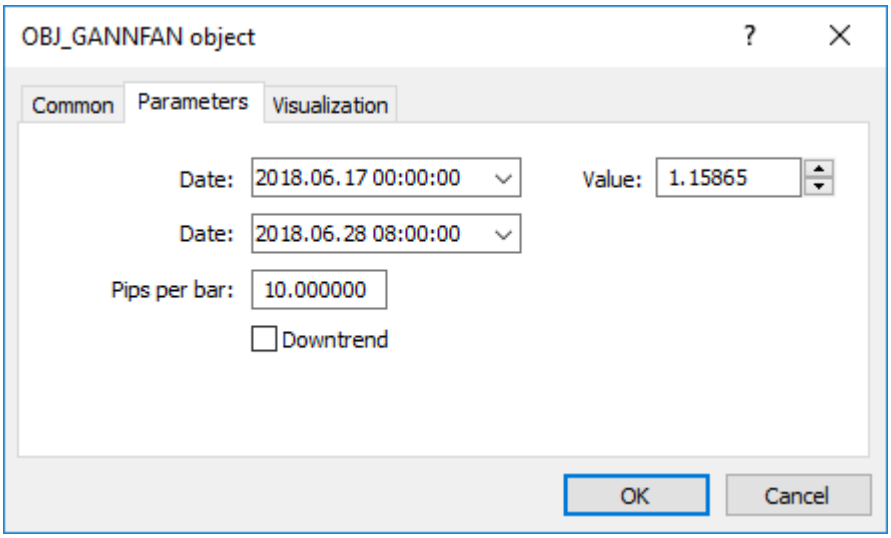
Gann Yelpazesi (OBJ_GANNFAN) ve Gann Izgarası (OBJ_GANNGRID) nesneleri için, trend yönünü ayarlayan ENUM_GANN_DIRECTION sayımından iki değer kullanabilirsiniz.

ENUM_GANN_DIRECTION

Tanıtıcı	Açıklama
GANN_UP_TREND	Yukarı yönlü trende karşılık gelen çizgi
GANN_DOWN_TREND	Aşağı yönlü trende karşılık gelen çizgi

Ana çizginin ölçeğini 1x1 olarak ayarlamak için, ObjectSetDouble(chart_handle, gann_object_name, OBJPROP_SCALE, scale) fonksiyonunu kullanın, burada:

- chart_handle - nesnenin konumlandırıldığı çizelge penceresidir;
- gann_object_name - nesne ismidir;
- OBJPROP_SCALE - "Scale" (ölçek) özelliğidir;
- scale - Pip/Çubuk birimleri bazında istenen ölçek değeridir.



Gann Yelpazesi oluşturma örneği:

```
void OnStart()
  {
//---
   string my_gann="OBJ_GANNFAN object";
   if(ObjectFind(0,my_gann)<0)// Nesne bulunamadı
     {
      //--- Hata hakkında bilgilendir
      Print("Nesne ",my_gann," bulunamadı. Hata kodu = ",GetLastError());
      //--- Çizelgedeki maksimal fiyat değerini al
      double chart_max_price=ChartGetDouble(0,CHART_PRICE_MAX,0);
      //--- Çizelgedeki minimal fiyat değerini al
      double chart_min_price=ChartGetDouble(0,CHART_PRICE_MIN,0);
      //--- Çizelgede kaç tane çubuk gösteriliyor?
```

```
      int bars_on_chart=ChartGetInteger(0,CHART_VISIBLE_BARS);
      //--- Her bir çubuğun açılış anını içeren bir dizi oluştur
      datetime Time[];
      //--- Diziye erişimi zaman serilerindeki gibi ayarla
      ArraySetAsSeries(Time,true);
      //--- Şimdi çizelgede görünür olan çubukların verilerini bu diziye kopyala
      int times=CopyTime(NULL,0,0,bars_on_chart,Time);
      if(times<=0)
        {
         Print("Açılış zamanlarını içeren dizi kopyalanamadı!");
         return;
        }
      //--- Ön hazırlıklar tamamlandı

      //--- Çizelgede ortadaki çubuğun indisi
      int center_bar=bars_on_chart/2;
      //--- Çizelge ortası - maksimum ve minimum arası
      double mean=(chart_max_price+chart_min_price)/2.0;
      //--- Ilk tutturma noktasının koordinatlarını merkeze ayarla
      ObjectCreate(0,my_gann,OBJ_GANNFAN,0,Time[center_bar],mean,
                   //--- İkinci tutturma noktası sağa
                   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
      //--- Ölçeği Pip / Çubuk birimleriyle ayarla
      ObjectSetDouble(0,my_gann,OBJPROP_SCALE,10);
      //--- Çizgi trendini ayarla
      ObjectSetInteger(0,my_gann,OBJPROP_DIRECTION,GANN_UP_TREND);
      //--- Çizgi genişliğini ayarla
      ObjectSetInteger(0,my_gann,OBJPROP_WIDTH,1);
      //--- Çizgi stilini tanımla
      ObjectSetInteger(0,my_gann,OBJPROP_STYLE,STYLE_DASHDOT);
      //--- Çizgi rengini ayarla
      ObjectSetInteger(0,my_gann,OBJPROP_COLOR,clrYellowGreen);
      //--- Kullanıcıya bir nesne seçmesi için izin ver
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTABLE,true);
      //--- Kendin seç
      ObjectSetInteger(0,my_gann,OBJPROP_SELECTED,true);
      //--- Bunu çizelge üzerine çiz
      ChartRedraw(0);
     }
  }
```

Web Renkleri

Aşağıdaki renk sabitleri color tipi için tanımlanmıştır:

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				

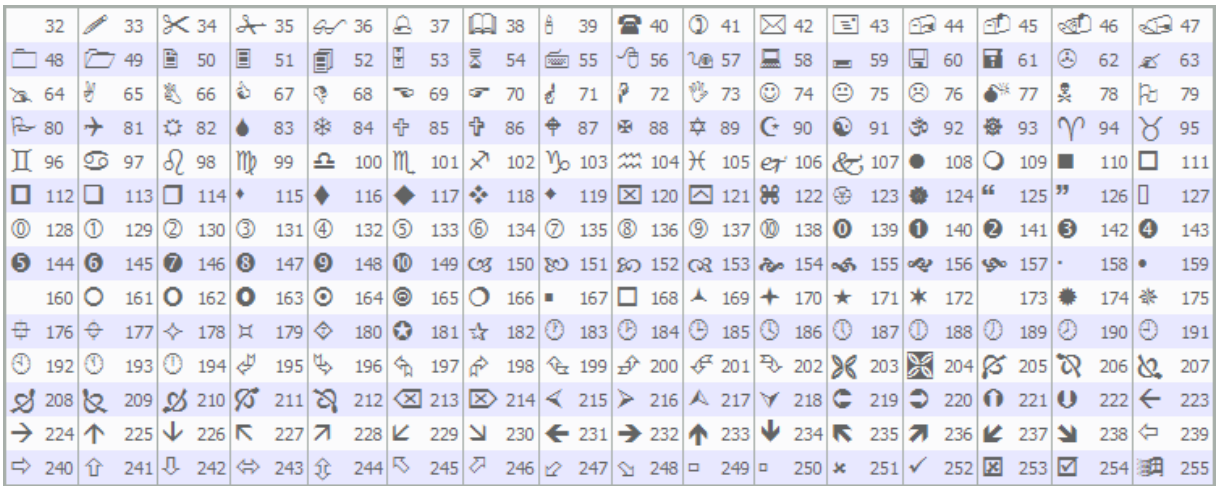
Nesne rengi ObjectSetInteger() fonksiyonuyla ayarlanabilir. Özel göstergelerde renk ayarlamak için PlotIndexSetInteger() fonksiyonu kullanılır. Renk değerlerinin elde edilmesi için de ObjectGetInteger() ve PlotIndexGetInteger() fonksiyonları bulunmaktadır.

Örnek:

```
//---- gösterge ayarları
#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

Wingdings karakterleri, OBJ_ARROW nesnesiyle birlikte kullanılırlar:



Gereken karakter ObjectSetInteger() fonksiyonuyla ayarlanabilir.

Örnek:

```
void OnStart()
  {
//---
   string up_arrow="up_arrow";
   datetime time=TimeCurrent();
   double lastClose[1];
   int close=CopyClose(Symbol(),Period(),0,1,lastClose);      // Kapanış (Close) fiyatı
//--- Eğer fiyat alındıysa
   if(close>0)
     {
      ObjectCreate(0,up_arrow,OBJ_ARROW,0,0,0,0,0);           // Bir ok oluştur
      ObjectSetInteger(0,up_arrow,OBJPROP_ARROWCODE,241);     // Okun kodunu ayarla
      ObjectSetInteger(0,up_arrow,OBJPROP_TIME,time);         // Zamanı ayarla
      ObjectSetDouble(0,up_arrow,OBJPROP_PRICE,lastClose[0]);// Fyatı ayarla
      ChartRedraw(0);                                         // Şimdi oku çiz
     }
   else
      Print("Son kapanış (Close) fiyatları alınamadı!");
  }
```

Gösterge Sabitleri

MQL5 dilinde program yazarken kullanılabilecek 37 adet ön tanımlı teknik gösterge bulunmaktadır. Buna ek olarak, iCustom() fonksiyonunun kullanımıyla özel göstergeler de oluşturulabilir. Bunun için gerekli olan tüm sabitler, 5 ayrı gruba bölünmüştür:

- Fiyat sabitleri – hesaplanan göstergede kullanılacak fiyat ve hacim tipini seçmek için kullanılır;
- Düzleştirme yöntemleri – göstergelerde kullanılmak amacıyla gömülü (kullanıma hazır) düzleştirme yöntemleri;
- Gösterge çizgileri – CopyBuffer() fonksiyonu ile gösterge değerlerine erişirken kullanılan gösterge tamponlarının tanıtıcıları;
- Çizim stilleri – var olan 18 çizim tipinden birinin belirtilmesi ve çizgi stilinin ayarlanması için kullanılır;
- Özel gösterge özellikleri - özel göstergelerle çalışan fonksiyonlarda kullanılır;
- Gösterge tipleri - IndicatorCreate() fonksiyonu kullanarak bir teknik göstergenin tanıtıcı değeri oluşturulurken, gösterge tipini belirtmek için kullanılır;
- Veri tipi tanımlayıcıları - IndicatorCreate() fonksiyonuna, MqlParam tipi bir dizi ile geçirilen verinin tipini belirtmek için kullanılır

Fiyat Sabitleri

Teknik göstergelerin hesaplanması, hesaplamada kullanılacak fiyat ve/veya hacim değerlerini gerektirir. Hesaplamada kullanılacak fiyat temelinin belirlenmesi için, ENUM_APPLIED_PRICE sayımı içinde 7 adet hazır tanıtıcı bulunmaktadır.

ENUM_APPLIED_PRICE

Tanıtıcı	Açıklama
PRICE_CLOSE	Kapanış fiyatı
PRICE_OPEN	Açılış fiyatı
PRICE_HIGH	Periyot içi maksimum fiyat
PRICE_LOW	Periyot içi maksimum fiyat
PRICE_MEDIAN	Medyan fiyat, (high + low)/2
PRICE_TYPICAL	Tipik fiyat, (high + low + close)/3
PRICE_WEIGHTED	Ağırlıklı ortalama fiyat, (high + low + close + close)/4

Eğer hesaplama için hacim değerleri kullanılacaksa, ENUM_APPLIED_VOLUME sayımındaki iki değerden birinin belirtilmesi gereklidir.

ENUM_APPLIED_VOLUME

Tanıtıcı	Açıklama
VOLUME_TICK	Tik hacmi
VOLUME_REAL	Alım-satım hacmi

iStochastic() teknik göstergesi şu iki yoldan biri ile hesaplanabilir:

- ya sadece Close fiyatları;
- ya da High ve Low fiyatları.

Hesaplamada kullanılmak istenilen gerekli bir varyantın seçilebilmesi için, ENUM_STO_PRICE sayımının değerlerinden biri belirtilmelidir.

ENUM_STO_PRICE

Tanıtıcı	Açıklama
STO_LOWHIGH	Low/High fiyatları temelinde hesaplama
STO_CLOSECLOSE	Close/Close fiyatları temelinde hesaplama

Eğer bir teknik gösterge, hesaplamalar için, tipi ENUM_APPLIED_PRICE ile ayarlanmış fiyat verisini kullanıyorsa, o zaman (terminalde kurulu veya bir kullanıcı tarafından yazılmış) herhangi bir göstergenin tanıtıcı değeri, giriş fiyat serisi olarak kullanılabilir. Bu durumda, hesaplamalar için sıfır tamponunun değerleri kullanılacaktır. Bu şekilde, bir göstergenin değerleri ile başka bir göstergenin kurulması kolaylaşır. Özel bir göstergenin tanıtıcı değeri iCustom() fonksiyonu kullanılarak oluşturulur.

Örnek:

```
#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- giriş parametreleri
input int      RSIperiod=14;         // RSI hesabı için periyot
input int      Smooth=8;             // RSI düzleştirme periyodu
input ENUM_MA_METHOD meth=MODE_SMMA; // Düzleştirme yöntemi
//---- plot RSI
#property indicator_label1  "RSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//---- plot 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
//--- gösterge tamponları
double         RSIBuffer[];          // Buraya RSI değerlerini depolayacağız
double         RSI_SmoothedBuffer[]; // Burada düzgünleştirilmiş RSI değerleri yer ala
int            RSIhandle;            // RSI göstergesinin tanıtıcı değeri
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   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);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[]
                )

  {
```

```
//--- Son hata değerini sıfırla
   ResetLastError();
//--- RSI gösterge verisini, RSIBuffer[] dizisinden al
   int copied=CopyBuffer(RSIhandle,0,0,rates_total,RSIBuffer);
   if(copied<=0)
     {
      Print("RSI göstergesinin değerleri kopyalanamıyor. Hata = ",
            GetLastError(),",  kopyalanan =",copied);
      return(0);
     }
//--- RSI değerlerini kullanarak, ortalama değerli göstergeyi oluştur
   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("Düzleştirilmiş RSI göstergesi kopyalanamıyor. Hata = ",
            GetLastError(),",  kopyalanan =",copied);
      return(0);
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Düzleştirme Yöntemleri

Bir çok teknik gösterge, fiyat serilerinin düzleştirildiği çeşitli yöntemler temel alınarak yazılmıştır. Bazı standart teknik göstergeler, giriş parametresi olarak, düzleştirme tipinin belirtilmesini gerektirir. İstenilen düzleştirme tipini belirtmek için, ENUM_MA_METHOD sayımında listelenen tanımlayıcılar kullanılır.

ENUM_MA_METHOD

Tanıtıcı	Açıklama
MODE_SMA	Basit ortalama
MODE_EMA	Üssel ortalama
MODE_SMMA	Düzleştirilmiş ortalama
MODE_LWMA	Doğrusal-ağırlıklı ortalama

Örnek:

```
double ExtJaws[];
double ExtTeeth[];
double ExtLips[];
//---- hareketli ortalamalar için işleyiciler
int    ExtJawsHandle;
int    ExtTeethHandle;
int    ExtLipsHandle;
//--- MA işleyicilerini al
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);
```

Gösterge Çizgileri

Bazı teknik göstergelerde çizelgede gösterilecek birkaç tampon bulunur. Gösterge tamponlarının indislenmesine 0'dan başlanır. Gösterge değerleri CopyBuffer() fonksiyonuyla double tipli bir diziye kopyalandığında, bazı göstergeler için bu kopyalanan tamponun numarasını değil tanıtıcısını belirtir.

iMACD(), iRVI() ve iStochastic() değerlerinin kopyalanması için gösterge çizgi tanıtıcılarının kullanımına izin verilir.

Sabit	Değer	Açıklama
MAIN_LINE	0	Ana çizgi
SIGNAL_LINE	1	Sinyal çizgisi

Gösterge çizgi tanıtıcılarının kullanımına, ADX() ve ADXW() değerlerinin kopyalanmasında izin verilir.

Sabit	Değer	Açıklama
MAIN_LINE	0	Ana çizgi
PLUSDI_LINE	1	Çizgi +DI
MINUSDI_LINE	2	Çizgi -DI

Gösterge çizgi tanıtıcılarının kullanımına, iBands() değerlerinin kopyalanmasında izin verilir.

Sabit	Değer	Açıklama
BASE_LINE	0	Ana çizgi
UPPER_BAND	1	Üst limit
LOWER_BAND	2	Alt limit

Gösterge çizgi tanıtıcılarının kullanımına, iEnvelopes() ve iFractals() değerlerinin kopyalanmasında izin verilir.

Sabit	Değer	Açıklama
UPPER_LINE	0	Üst çizgi
LOWER_LINE	1	Alt çizgi

Gösterge çizgi tanıtıcılarının kullanımına, iGator() değerlerinin kopyalanmasında izin verilir.

Sabit	Değer	Açıklama
UPPER_HISTOGRAM	0	Üst histogram
LOWER_HISTOGRAM	2	Alt histogram

Gösterge çizgi tanıtıcılarının kullanımına, iAlligator() değerlerinin kopyalanmasında izin verilir.

Sabit	Değer	Açıklama
GATORJAW_LINE	0	Çene çizgisi
GATORTEETH_LINE	1	Diş çizgisi
GATORLIPS_LINE	2	Dudak çizgisi

Gösterge çizgi tanıtıcılarının kullanımına, iIchimoku() değerlerinin kopyalanmasında izin verilir.

Sabit	Değer	Açıklama
TENKANSEN_LINE	0	Tenkan-sen çizgisi
KIJUNSEN_LINE	1	Kijun-sen çizgisi
SENKOUSPANA_LINE	2	Senkou Span A çizgisi
SENKOUSPANB_LINE	3	Senkou Span B çizgisi
CHIKOUSPAN_LINE	4	Chikou Span çizgisi

Çizim Stilleri

Bir özel gösterge oluştururken, değerleri ENUM_DRAW_TYPE sayımı içinde belirlenen 18 grafiksel çizim tipinden birini (ana çizelge penceresinde veya alt pencerede görüntülendiği şekilde) belirleyebilirsiniz.

Bir özel gösterge içinde herhangi bir gösterge kurma/çizme tipini kullanabilirsiniz. Her bir çizim tipi, bir ile beş adet global dizinin belirtilmesini gerektirir. Bu veri dizileri SetIndexBuffer() fonksiyonu kullanılarak, gösterge tamponlarına bağlanmalıdır. Her bir tampon için, ENUM_INDEXBUFFER_TYPE sayımı içinden bir veri tipi belirlenmelidir.

Çizim stiline bağlı olarak, (INDICATOR_DATA şeklinde imlenmiş) bir ile dört değer tamponuna ihtiyaç duyabilirsiniz. Eğer bir stil, renklerin dinamik değişimini içeriyorsa (tüm stiller COLOR sözcüğünü içerir), diğer renk için bir tampona daha ihtiyacınız olacaktır (istenen tip INDICATOR_COLOR_INDEX). Renk tamponları her zaman stil rengine karşılık gelen değer tamponundan sonra bağlanırlar.

ENUM_DRAW_TYPE

Tanıtıcı	Açıklama	Veri tamponları	Renk tamponları
DRAW_NONE	Çizilmez	1	0
DRAW_LINE	Çizgi	1	0
DRAW_SECTION	Bölüm	1	0
DRAW_HISTOGRAM	Sıfır çizgisinden başlayan histogram	1	0
DRAW_HISTOGRAM2	İki gösterge tamponunun histogramı	2	0
DRAW_ARROW	Ok çizimi	1	0
DRAW_ZIGZAG	Zigzag stili, çubuklar için dikey kesitler oluşturulmasını sağlar	2	0
DRAW_FILLING	İki seviye arasının	2	0

Tanıtıcı	Açıklama	Veri tamponları	Renk tamponları
	renk ile doldurulması		
DRAW_BARS	Çubuk dizilimi şeklinde gösterir	4	0
DRAW_CANDLES	Mumların dizilimi şeklinde gösterir	4	0
DRAW_COLOR_LINE	Çok renkli çizgi	1	1
DRAW_COLOR_SECTION	Çok renkli kesit	1	1
DRAW_COLOR_HISTOGRAM	Sıfır çizgisinden başlayan çok renkli histogram	1	1
DRAW_COLOR_HISTOGRAM2	İki gösterge tamponunun çok renkli histogramı	2	1
DRAW_COLOR_ARROW	Çok renkli okların çizimi	1	1
DRAW_COLOR_ZIGZAG	Çok renkli ZigZag	2	1
DRAW_COLOR_BARS	Çok renkli çubuklar	4	1

Tanıtıcı	Açıklama	Veri tamponları	Renk tamponları
DRAW_COLOR_CANDLES	Çok renkli mumlar	4	1

Seçilen çizim tipinin görümünü düzenlemek için ENUM_PLOT_PROPERTY özelliğindeki tanımlayıcılar kullanılır.

PlotIndexSetInteger() ve PlotIndexGetInteger() fonksiyonları için

ENUM_PLOT_PROPERTY_INTEGER

Tanıtıcı	Açıklama	Özellik tipi
PLOT_ARROW	DRAW_ARROW stili için ok kodu	uchar
PLOT_ARROW_SHIFT	DRAW_ARROW stilinde okların dikey olarak kaydırılması	int
PLOT_DRAW_BEGIN	Çizim yapılmamış çubuk sayısı ve Veri Penceresindeki değerler	int
PLOT_DRAW_TYPE	Grafiksel çizim tipi	ENUM_DRAW_TYPE
PLOT_SHOW_DATA	Çizim değerlerinin Veri Penceresinde görüntülenen işareti	bool
PLOT_SHIFT	Gösterge grafiğinin, zaman ekseninde çubuk bazında kaydırılması	int
PLOT_LINE_STYLE	Çizgi çizim stili	ENUM_LINE_STYLE
PLOT_LINE_WIDTH	Çizgi çizim kalınlığı	int
PLOT_COLOR_INDEXES	Renk sayısı	int
PLOT_LINE_COLOR	Çizim rengini içeren bir tamponun indisi	color şekillendirici = renklerin indis numaraları

PlotIndexSetDouble() fonksiyonu için

ENUM_PLOT_PROPERTY_DOUBLE

Tanıtıcı	Açıklama	Özellik tipi
PLOT_EMPTY_VALUE	Çizim yapılmayan bir grafik için boş değer	double

PlotIndexSetString() fonksiyonu için

ENUM_PLOT_PROPERTY_STRING

Tanıtıcı	Açıklama	Özellik tipi
PLOT_LABEL	Veri Penceresinde gösterilecek grafiksel serinin ismi. Birkaç gösterge tamponunun kullanımını gerektiren karmaşık grafik stilleri ile çalışırken, her bir tamponun ismi ";" ayracı kullanılarak belirtilebilir. Örnek kod şurada gösterilmektedir DRAW_CANDLES	string

Özel göstergelerde, çizim amaçlı 5 farklı stil kullanılabilir. Bunlar sadece 0 ve 1 çizgi kalınlıklarında geçerlidir.

ENUM_LINE_STYLE

Tanıtıcı	Açıklama
STYLE_SOLID	Düz çizgi
STYLE_DASH	Kesikli çizgi
STYLE_DOT	Noktalı çizgi
STYLE_DASHDOT	Kesikli-noktalı çizgi
STYLE_DASHDOTDOT	Kesikli - çift noktalı çizgi

Çizim stilini ve tipini ayarlamak için, PlotIndexSetInteger() fonksiyonu kullanılır. Fibonacci açılımı için kalınlık ve çizim stili ObjectSetInteger() fonksiyonu kullanılarak belirtilebilir.

Örnek:

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- gösterge tamponları
double         MABuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- Diziyi, 0 indisli gösterge tamponuna bağlar
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
//--- Çizim tipini ayarla
   PlotIndexSetInteger(0,PLOT_DRAW_TYPE,DRAW_LINE);
//--- Çizgi stilini ayarla
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,STYLE_DOT);
//--- Çizgi rengini ayarla
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrRed);
//--- Çizgi kalınlığını ayarla
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
//--- Çizgiler için etiket ayarla
   PlotIndexSetString(0,PLOT_LABEL,"Moving Average");
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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];
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Özel Göstergelerin Özellikleri

Bir özel gösterge içinde kullanılabilecek gösterge tamponlarının sayısı sınırsızdır. SetIndexBuffer() fonksiyonu ile bir gösterge tamponu şeklinde tasarlanmış her bir dizi için, depolanacak veri tipinin belirtilmesi gerekir. Bu, ENUM_INDEXBUFFER_TYPE sayımının değerlerinden biri olabilir.

ENUM_INDEXBUFFER_TYPE

Tanıtıcı	Açıklama
INDICATOR_DATA	Çizilecek veri
INDICATOR_COLOR_INDEX	Renk
INDICATOR_CALCULATIONS	Ara işlemler için kullanılacak yardımcı tamponlar

Özel göstergeler, uygun görüntülemenin sağlanması için bir çok ayarlama içerir. Bu ayarlamalar, IndicatorSetDouble(), IndicatorSetInteger() ve IndicatorSetString() fonksiyonları kullanılarak, karşılık gelen gösterge özelliklerinin atanmasıyla gerçekleştirilir. Gösterge özelliklerinin tanıtıcıları, ENUM_CUSTOMIND_PROPERTY sayımında listelenmiştir.

ENUM_CUSTOMIND_PROPERTY_INTEGER

Tanıtıcı	Açıklama	Özellik tipi
INDICATOR_DIGITS	Gösterge değerlerinin çiziminin doğruluğu	int
INDICATOR_HEIGHT	Sabitlenmiş göstergepencere si yüksekliği (önişle	int

Tanıtıcı	Açıklama	Özellik tipi
	mci komutu #property indicator_height)	
INDICATOR_LEVELS	Gösterge penceresindeki seviyelerin sayısı	int
INDICATOR_LEVELCOLOR	Seviye çizgisinin rengi	color şekillendirici = seviye numarası
INDICATOR_LEVELSTYLE	Seviye çizgisinin stili	ENUM_LINE_STYLE şekillendirici = seviye numarası
INDICATOR_LEVELWIDTH	Seviye çizgisinin kalınlığı	int şekillendirici = seviye numarası

Tanıtıcı	Açıklama	Özellik tipi
INDICATOR_FIXED_MINIMUM	Gösterge penceresi için sabit minimum. Özellik yalnızca IndicatorSetInteger() fonksiyonu tarafından yazılabilir	bool
INDICATOR_FIXED_MAXIMUM	Gösterge penceresi için sabit maksimum. Özellik yalnızca	bool

Tanıtıcı	Açıklama	Özellik tipi
	IndicatorSetInteger() fonksiyonu tarafından yazılabilir	

ENUM_CUSTOMIND_PROPERTY_DOUBLE

Tanıtıcı	Açıklama	Özellik tipi
INDICATOR_MINIMUM	Gösterge penceresinin minimum değeri	double
INDICATOR_MAXIMUM	Gösterge penceresinin maksimum değeri	double

Tanıtıcı	Açıklama	Özellik tipi
INDICATOR_LEVELVALUE	Seviye değeri	double şekillendirici = seviye numarası

ENUM_CUSTOMIND_PROPERTY_STRING

Tanıtıcı	Açıklama	Özellik tipi
INDICATOR_SHORTNAME	Gösterge kısa ismi	string
INDICATOR_LEVELTEXT	Seviye açıklaması	string şekillendirici = seviye numarası

Örnekler:

```
//--- gösterge ayarları
#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
//--- giriş parametreleri
extern int KPeriod=5;
extern int DPeriod=3;
extern int Slowing=3;
//--- gösterge tamponları
double MainBuffer[];
double SignalBuffer[];
double HighesBuffer[];
double LowesBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
```

```
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,MainBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,HighesBuffer,INDICATOR_CALCULATIONS);
   SetIndexBuffer(3,LowesBuffer,INDICATOR_CALCULATIONS);
//--- kesinliği ayarla
   IndicatorSetInteger(INDICATOR_DIGITS,2);
//--- seviyeleri ayarla
   IndicatorSetInteger(INDICATOR_LEVELS,2);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,0,20);
   IndicatorSetDouble(INDICATOR_LEVELVALUE,1,80);
//--- alt pencere için maksimum ve minimum değerleri ayarla
   IndicatorSetDouble(INDICATOR_MINIMUM,0);
   IndicatorSetDouble(INDICATOR_MAXIMUM,100);
//--- çizime hangi çubuğun indisinden başlanacağını ayarlar
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,KPeriod+Slowing-2);
   PlotIndexSetInteger(1,PLOT_DRAW_BEGIN,KPeriod+Slowing+DPeriod);
//--- ikinci çizgi için STYLE_DOT stilini ayarla
   PlotIndexSetInteger(1,PLOT_LINE_STYLE,STYLE_DOT);
//--- Veri Penceresi ve gösterge alt penceresinin etiketi için isim
   IndicatorSetString(INDICATOR_SHORTNAME,"Stoch("+KPeriod+","+DPeriod+","+Slowing+")"
   PlotIndexSetString(0,PLOT_LABEL,"Main");
   PlotIndexSetString(1,PLOT_LABEL,"Signal");
//--- çizimi boş değer olarak ayarlar
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0.0);
//--- başlatma tamamlandı
  }
```

Teknik Gösterge Tipleri

Gösterge tanıtıcı değerini, ileride değerlerine erişmek amacıyla programlanabilir şekilde oluşturmak için iki yol bulunmaktadır. İlk yol, teknik göstergeler listesinden bir fonksiyon ismini direk olarak belirtmektir. IndicatorCreate() fonksiyonunu kullanan ikinci yol ise, ENUM_INDICATOR sayımından bir tanımlayıcı atayarak, bir göstergenin tanıtıcı değeri düzgün şekilde oluşturmaktır. İki yöntem de birbirine eşdeğerdir; MQL5 diliyle bir program yazarken, bunlardan duruma en uygun olanı kullanabilirsiniz.

IND_CUSTOM tipinde bir gösterge oluşturulurken, MqlParam giriş parametreleri dizisinin ilk elemanındaki *type* alanı, ENUM_DATATYPE sayımının TYPE_STRING değerini, *string_value* alanı ise özel göstergenin ismini içermelidir.

ENUM_INDICATOR

Tanımlayıcı	Gösterge
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	Özel gösterge
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
IND_GATOR	Gator Oscillator

Tanımlayıcı	Gösterge
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

Veri Tipi Tanımlayıcıları

IndicatorCreate() fonksiyonunu kullanarak bir gösterge tanıtıcı değeri oluşturulurken, son parametre olarak MqlParam tipli bir dizi belirtilmelidir. Buna göre, gösterge parametrelerini tarif eden MqlParam yapısı, *type* isminde özel bir alan içerir. Bu alan, dizinin belirli bir elemanına aktarılan veri tipi (reel, tamsayı veya string tipi) hakkında bilgi içerir. MqlParam yapısının bu alanının değeri, ENUM_DATATYPE sayımının değerlerinden biri olabilir.

ENUM_DATATYPE

Tanımlayıcı	Veri Tipi
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

Dizinin her bir elemanı, oluşturulan teknik göstergenin karşılık gelen giriş parametresini tarif eder; dizideki elemanların tipleri ve sıraları, bu tarife sıkıca uymalıdır.

Ortam Durumu

Bir MQL5 programının çalışma ortamını tarif eden sabitler gruplara bölünür:

- Müşteri terminali özellikleri – müşteri terminali hakkında bilgi;
- Çalıştırılmış MQL5 programı özellikleri – MQL5 programının çalıştırılmasına yardımcı olan özellikler;
- Sembol Özellikleri – bir sembol hakkında bilgi edinme;
- Hesap Özellikleri – mevcut hesap hakkında bilgi;
- Sınama İstatistikleri – Uzman Danışmanın sınama sonuçları.

Müşteri Terminali Özellikleri

Müşteri terminali hakkında bilgi almak için iki fonksiyon bulunmaktadır: TerminalInfoInteger() ve TerminalInfoString(). Fonksiyon parametreleri sırasıyla ENUM_TERMINAL_INFO_INTEGER ve ENUM_TERMINAL_INFO_STRING sayımlarının değerlerini alır.

ENUM_TERMINAL_INFO_INTEGER

Tanımlayıcı	Açıklama	Tip
TERMINAL_BUILD	Müşteri terminalinin kurulum numarası	int
TERMINAL_COMMUNITY_ACCOUNT	Bayrak, terminalde MQL5.community yetkilendirme verisinin bulunduğunu belirtir.	bool
TERMINAL_COMMUNITY_CONNECTION	MQL5.community'e bağlanma	bool
TERMINAL_CONNECTED	Bir alım-satım sunucusuna yapılan bağlantı	bool
TERMINAL_DLLS_ALLOWED	DLL kullanım izni	bool
TERMINAL_TRADE_ALLOWED	Alım-satım yapma izni	bool
TERMINAL_EMAIL_ENABLED	SMTP sunucusu ve terminal ayarlarında belirtilmiş giriş bilgisini (login) kullanılarak, e-mail gönderme izni	bool
TERMINAL_FTP_ENABLED	FTP sunucusu ve terminal ayarlarında belirtilmiş giriş bilgisini (login) kullanılarak, rapor gönderme izni	bool
TERMINAL_NOTIFICATIONS_ENABLED	Akıllı telefona bildirim gönderme izni	bool
TERMINAL_MAXBARS	Çizelge üzerindeki maksimum çubuk sayısı	int
TERMINAL_MQID	Bayrak Push bildirimlerini göndermek için MetaQuotes ID verilerinin varlığını gösterir.	bool
TERMINAL_CODEPAGE	Müşteri terminalinde yüklü olan dilin kod sayfasının numarası	int
TERMINAL_CPU_CORES	Sistemdeki CPU (işlemci) çekirdeklerinin sayısı	int
TERMINAL_DISK_SPACE	Terminal biriminin MQL5\Files klasörü için kullanılabilecek serbest disk alanı, MB	int

Tanımlayıcı	Açıklama	Tip
TERMINAL_MEMORY_PHYSICAL	Sistemdeki mevcut fiziksel bellek, MB	int
TERMINAL_MEMORY_TOTAL	Terminal birimi işlemleri için kullanılabilir bellek büyüklüğü, MB	int
TERMINAL_MEMORY_AVAILABLE	Terminal birimi işlemlerindeki serbest bellek, MB	int
TERMINAL_MEMORY_USED	Temsilci tarafından kullanılan bellek miktarı, MB	int
TERMINAL_X64	"64-bitlik terminal" gösterimi	bool
TERMINAL_OPENCL_SUPPORT	Desteklenen OpenCL versiyonu, 0x00010002 = 1.2. biçiminde verilir "0" sayısı OpenCL'in desteklenmediği anlamına gelir	int
TERMINAL_SCREEN_DPI	Ekran görüntüleme çözünürlüğü, ekran üzerindeki bir inçlik çizgideki noktaların sayısıyla (Dots Per Inch – DPI) ölçülür. Bu parametrenin değerini bilirseniz, grafiksel nesnelerin boyutlarını farklı çözünürlük özelliklerine sahip ekranlarda aynı görüntülenecek şekilde ayarlayabilirsiniz.	int
TERMINAL_SCREEN_LEFT	Sanal ekranın sol koordinatı. Sanal ekran tüm monitörleri kaplayan bir diktörtgendir. Eğer sistem sağdan sola dizili iki monitöre sahipse, o zaman sanal ekranın sol koordinatı iki monitörün sınırında bulunabilir.	int
TERMINAL_SCREEN_TOP	Sanal ekranın üst koordinatı	int
TERMINAL_SCREEN_WIDTH	Terminal genişlik	int
TERMINAL_SCREEN_HEIGHT	Terminal yükseklik	int
TERMINAL_LEFT	Sanal ekrana göre terminalin sol koordinatı	int
TERMINAL_TOP	Sanal ekrana göre terminalin üst koordinatı	int
TERMINAL_RIGHT	Sanal ekrana göre terminalin sağ koordinatı	int
TERMINAL_BOTTOM	Sanal ekrana göre terminalin alt koordinatı	int

Tanımlayıcı	Açıklama	Tip
TERMINAL_PING_LAST	Alım-satım sunucusuna gönderilen son ping sinyalinin mikrosaniye cinsinden değeri. Bir saniye, bir milyon mikrosaniyeden oluşur	int
TERMINAL_VPS	Terminalin MetaTrader Sanal Hosting sunucusunda (MetaTrader VPS) başlatıldığını gösterir.	bool
Anahtar Tanıtıcısı	**Açıklama**	
TERMINAL_KEYSTATE_LEFT	"Sol ok" tuşunun durumu	int
TERMINAL_KEYSTATE_UP	"Yukarı ok" tuşunun durumu	int
TERMINAL_KEYSTATE_RIGHT	"Sağ ok" tuşunun durumu	int
TERMINAL_KEYSTATE_DOWN	"Aşağı ok" tuşunun durumu	int
TERMINAL_KEYSTATE_SHIFT	"Shift" tuşunun durumu	int
TERMINAL_KEYSTATE_CONTROL	"Ctrl" tuşunun durumu	int
TERMINAL_KEYSTATE_MENU	"Windows" tuşunun durumu	int
TERMINAL_KEYSTATE_CAPSLOCK	"CapsLock" tuşunun durumu	int
TERMINAL_KEYSTATE_NUMLOCK	"NumLock" tuşunun durumu	int
TERMINAL_KEYSTATE_SCRLOCK	"ScrollLock" tuşunun durumu	int
TERMINAL_KEYSTATE_ENTER	"Enter" tuşunun durumu	int
TERMINAL_KEYSTATE_INSERT	"Insert" tuşunun durumu	int
TERMINAL_KEYSTATE_DELETE	"Delete" tuşunun durumu	int
TERMINAL_KEYSTATE_HOME	"Home" tuşunun durumu	int
TERMINAL_KEYSTATE_END	"End" tuşunun durumu	int
TERMINAL_KEYSTATE_TAB	"Tab" tuşunun durumu	int
TERMINAL_KEYSTATE_PAGEUP	"PageUp" tuşunun durumu	int
TERMINAL_KEYSTATE_PAGEDOWN	"PageDown" tuşunun durumu	int
TERMINAL_KEYSTATE_ESCAPE	"Escape" tuşunun durumu	int

TerminalInfoInteger(TERMINAL_KEYSTATE_XXX) çağrısı MSDN'deki GetKeyState() fonksiyonuyla aynı durum koduna dönüş yapar.

Örnek - ölçekleme faktörünün hesaplanması:

```
//--- Ekranda 1.5 inç genişliğinde bir düğme oluşturma
int screen_dpi = TerminalInfoInteger(TERMINAL_SCREEN_DPI); // Kullanıcı monitörünün DI
```

```
int base_width = 144;                                    // DPI=96 değerine sahip st
int width      = (button_width * screen_dpi) / 96;       // Kullanıcı monitörü (özel
...

//--- Ölçekleme faktörünün yüzdelik olarak hesaplanması
int scale_factor=(TerminalInfoInteger(TERMINAL_SCREEN_DPI) * 100) / 96;
//--- Ölçekleme faktörünün yüzdelik olarak hesaplanması
width=(base_width * scale_factor) / 100;
```

Yukarıdaki örnekteki grafiksel kaynak farklı özelliklere sahip monitörlerde aynı görünecektir. Kontrol bileşenlerinin (düğmeler, dialog pencereleri, vb.) boyutları kişiselleştirme ayarlarına karşılık gelir.

ENUM_TERMINAL_INFO_DOUBLE

Tanımlayıcı	Açıklama	Tip
TERMINAL_COMMUNITY_BALANCE	MQL5.community'deki bakiye	double
TERMINAL_RETRANSMISSION	Verilen bilgisayarda çalışan tüm uygulamalar ve hizmetler için TCP/IP protokolündeki yeniden gönderilen ağ paketlerinin yüzdesi. Paket kaybı en hızlı ve doğru yapılandırılmış ağlarda bile gerçekleşir. Bu durumda, alıcı ile gönderen arasında paket iletiminin onaylanması yoktur, bu nedenle kaybedilen paketler yeniden gönderilir. Yüzde, sistem ve arka plan etkinliği dahil olmak üzere tüm ağ etkinliği için hesaplandığı için, belirli bir terminal ile bir trade sunucusu arasındaki bağlantı kalitesinin bir göstergesi değildir. TERMINAL_RETRANSMISSION değeri, işletim sisteminden dakikada bir kez istenir. Terminalin kendisi bu değeri hesaplamaz.	double

Dosya işlemleri sadece iki dosya konumu içinde gerçekleştirilebilir; karşılık gelen dosya adresleri, TERMINAL_DATA_PATH ve TERMINAL_COMMONDATA_PATH özelikleri için istekler kullanılarak elde edilebilir.

ENUM_TERMINAL_INFO_STRING

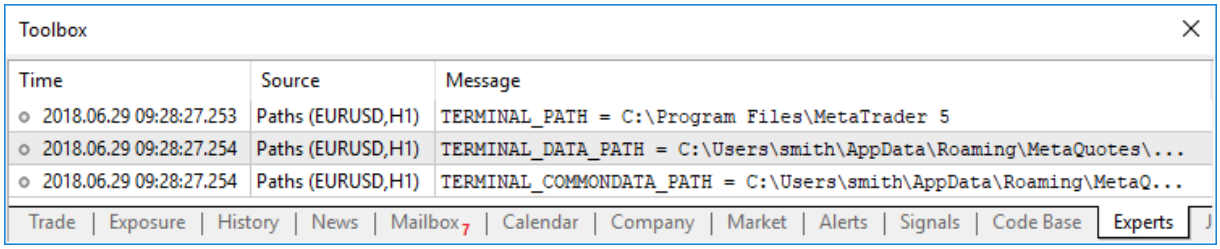
Tanımlayıcı	Açıklama	Tip
TERMINAL_LANGUAGE	Terminal dili	string
TERMINAL_COMPANY	Firma ismi	string
TERMINAL_NAME	Terminal ismi	string
TERMINAL_PATH	Terminalin başlatıldığı klasör	string
TERMINAL_DATA_PATH	Terminal verisinin saklandığı klasör	string
TERMINAL_COMMONDATA_PATH	Bilgisayara yüklenen tüm terminaller için çoğunlukla kullanılan dosya adresi	string
TERMINAL_CPU_NAME	CPU adı	string
TERMINAL_CPU_ARCHITECTURE	CPU mimarisi	string
TERMINAL_OS_VERSION	Kullanıcının işletim sistemi adı	string

TERMINAL_PATH, TERMINAL_DATA_PATH ve TERMINAL_COMMONDATA_PATH parametrelerinin özelliklerinde saklanan dosya yollarını daha iyi anlayabilmek için, bilgisayarınızda yüklü olan mevcut müşteri terminali kopyası için bu değerlere dönüş yapan bir betiği çalıştırmanız önerilir.

Örnek: Müşteri terminalinin dosya adresleri hakkında bilgi veren script

```
//+------------------------------------------------------------------+
//|                                          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"
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   Print("TERMINAL_PATH = ",TerminalInfoString(TERMINAL_PATH));
   Print("TERMINAL_DATA_PATH = ",TerminalInfoString(TERMINAL_DATA_PATH));
   Print("TERMINAL_COMMONDATA_PATH = ",TerminalInfoString(TERMINAL_COMMONDATA_PATH));
  }
```

Betiğin çalıştırılmasının sonucu olarak, Uzmanlar Günlüğünde şuna benzer bir mesaj göreceksiniz:



MQL5 Program Özelliklerini çalıştırma

Çalışmakta olan MQL5 programı hakkında bilgi edinmek için, ENUM_MQL_INFO_INTEGER ve ENUM_MQL_INFO_STRING ten sabitler kullanılır.

MQLInfoString fonksiyonu için

ENUM_MQL_INFO_INTEGER

Tanımlayıcı	Açıklama	Tip
MQL_HANDLES_USED	Mevcut etkin nesne tanıtıcısı sayısı. Bunlara hem dinamik (new aracılığıyla oluşturulan) hem de dinamik olmayan nesneler, global/yerel değişkenler veya sınıf üyeleri dahildir. Bir program ne kadar çok tanıtıcı kullanırsa, o kadar fazla kaynak tüketir.	int
MQL_MEMORY_LIMIT	MQL5 programı için dinamik belleğin MB cinsinden maksimum mümkün miktarı	int
MQL_MEMORY_USED	MQL5 programı tarafından kullanılan belleğin MB cinsinden miktarı	int
MQL_PROGRAM_TYPE	MQL5 programının tipi	ENUM_PROGRAM_TYPE
MQL_DLLS_ALLOWED	**Söz konusu çalışmakta olan program için** DLL kullanım izni	bool
MQL_TRADE_ALLOWED	**Söz konusu çalışmakta olan program için** alım-satım izni	bool

Tanımlayıcı	Açıklama	Tip
MQL_SIGNALS_ALLOWED	**Söz konusu çalışmakta olan program için** Sinyallerde değişiklik yapma izni	bool
MQL_DEBUG	Programın hata ayıklama modunda çalışmakta olduğunun işareti	bool
MQL_PROFILER	Programın kod profil oluşturma modunda çalışmakta olduğunun işareti	bool
MQL_TESTER	Programın test sınayıcısında çalışmakta olduğunun işareti	bool
MQL_FORWARD	Programın ileri test işleminde çalışmakta olduğunun işareti	bool
MQL_OPTIMIZATION	Programın optimizasyon modunda çalışmakta olduğunun işareti	bool
MQL_VISUAL_MODE	Programın görsel test modunda çalışmakta olduğunun işareti	bool
MQL_FRAME_MODE	Uzman Danışmanın optimizasyon sonuç çerçevelerinin toplanması modunda çalışmakta olduğunun işareti	bool
MQL_LICENSE_TYPE	EX5 modülünün lisans tipi.	ENUM_LICENSE_TYPE

Tanımlayıcı	Açıklama	Tip
	Lisans, Mql5InfoInteger(MQL_LICENSE_TYPE) kullanılarak bir isteğin yapıldığı EX5 modülünü ifade eder.	

MQLInfoString fonksiyonu için

ENUM_MQL_INFO_STRING

Tanımlayıcı	Açıklama	Tip
MQL_PROGRAM_NAME	Çalışmakta olan MQL5 programının ismi	string
MQL5_PROGRAM_PATH	**Söz konusu çalışmakta olan program için** dosya yolu	string

Çalışmakta olan programın tipi hakkında bilgi edinmek amacıyla, ENUM_PROGRAM_TYPE ın değerleri kullanılır.

ENUM_PROGRAM_TYPE

Tanımlayıcı	Açıklama
PROGRAM_SCRIPT	Komut dosyası
PROGRAM_EXPERT	Uzman danışman
PROGRAM_INDICATOR	Gösterge
PROGRAM_SERVICE	Hizmet

ENUM_LICENSE_TYPE

Tanımlayıcı	Açıklama
LICENSE_FREE	Ücretsiz sınırsız versiyon
LICENSE_DEMO	Market'ten ücretli bir ürünün deneme versiyonu. Sadece strateji sınayıcısında çalışır
LICENSE_FULL	Satın alınmış lisanslı bir versiyon en az 5 aktifleştirmeye izin verir. Aktifleştirme sayısı satıcı

Tanımlayıcı	Açıklama
	tarafından belirlenir. Satıcı izin verilen aktifleştirme sayısını arttırabilir
LICENSE_TIME	Sınırlı süreli lisansa sahip bir versiyon

Örnek:

```
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
```

Sembol Özellikleri

Mevcut piyasa bilgisini almak için birkaç fonksiyon bulunmaktadır: SymbolInfoInteger(), SymbolInfoDouble() ve SymbolInfoString(). İlk parametre sembol ismidir, ikinci fonksiyon parametresinin değerleri ENUM_SYMBOL_INFO_INTEGER, ENUM_SYMBOL_INFO_DOUBLE ve ENUM_SYMBOL_INFO_STRING tanıtıcılarından biri olabilir.

SymbolInfoInteger() fonksiyonu için

ENUM_SYMBOL_INFO_INTEGER

Tanıtıcı	Açıklama	Tip
SYMBOL_SECTOR	Varlığın ait olduğu ekonomi sektörü	ENUM_SYMBOL_SECTOR
SYMBOL_INDUSTRY	Finansal sembolün ait olduğu sanayi veya ekonomi dalı	ENUM_SYMBOL_INDUSTRY
SYMBOL_CUSTOM	Kullanıcı-tanımlı sembol - Piyasa	bool

Tanıtıcı	Açıklama	Tip
	Gözlemi pencere sindeki sembolleri n ve/veya dışsal kaynakların verilerinden yapay olarak oluşturulan özel bir sembold ür	
SYMBOL_BACKGROUND_COLOR	Piyasa Gözlemi ndeki sembol için kullanılan arkaplan rengi	color

Tanıtıcı	Açıklama	Tip
SYMBOL_CHART_MODE	Sembolün çubuk verilerini oluşturmak için kullanılacak fiyat tipi (satış veya son fiyat)	ENUM_SYMBOL_CHART_MODE
SYMBOL_EXIST	Aynı isimde bir sembol bulunmaktadı	bool
SYMBOL_SELECT	Piyasa Gözlemi penceresinde seçilen	bool

Tanıtıcı	Açıklama	Tip
	sembol	
SYMBOL_VISIBLE	Sembol Piyasa Gözleminde görünür. Bazı semboller (çoğunlukla bunlar temin at gereksinimlerin ve mevduat döviz cinsinden karın hesaplanması için kullanılır) otomat	bool

Tanıtıcı	Açıklama	Tip
	ik olarak seçilir ama genellikle Piyasa Gözleminde görünür değildir. Görüntülenecek semboller açıkça seçilmelidir.	
SYMBOL_SESSION_DEALS	Mevcut seans için deki işlemlerin sayısı	long
SYMBOL_SESSION_BUY_ORDERS	Mevcut Alım emirleri	long

Tanıtıcı	Açıklama	Tip
	nin sayısı	
SYMBOL_SESSION_SELL_ORDERS	Mevcut Satış emirlerinin sayısı	long
SYMBOL_VOLUME	Son işlemin hacmi	long
SYMBOL_VOLUMEHIGH	Günün en yüksek hacim değeri	long
SYMBOL_VOLUMELOW	Günün en düşük hacim değeri	long
SYMBOL_TIME	Son fiyat teklifinin zamanı	datetime

Tanıtıcı	Açıklama	Tip
SYMBOL_TIME_MSC	Son fiyatın 1970.01.01'den beri milisaniye cinsinden zamanı	long
SYMBOL_DIGITS	Ondalık noktadan sonraki basamak sayısı	int
SYMBOL_SPREAD_FLOAT	Değişken maks göstergesi mi	bool
SYMBOL_SPREAD	Puan bazında maks değeri	int

Tanıtıcı	Açıklama	Tip
SYMBOL_TICKS_BOOKDEPTH	Piyasa derinliğinde gösterilen isteklerin maksimal sayısı. İstek kuyruğu bulunmayan sembollerde, bu değer varsayılan olarak sıfırdır.	int
SYMBOL_TRADE_CALC_MODE	İşlem fiyatı hesaplama modu	ENUM_SYMBOL_CALC_MODE

Tanıtıcı	Açıklama	Tip
SYMBOL_TRADE_MODE	Emir gerçekleştirme tipi	ENUM_SYMBOL_TRADE_MODE
SYMBOL_START_TIME	Sembol alım-satım başlangıç tarihi (genellikle vadeli işlemlerde kullanılır)	datetime
SYMBOL_EXPIRATION_TIME	Sembol alım-satım bitiş tarihi (genellikle vadeli işlemlerde	datetime

Tanıtıcı	Açıklama	Tip
	kullanılır)	
SYMBOL_TRADE_STOPS_LEVEL	Stop emrinin verilebilmesi için, mevcut kapanış fiyatından itibaren, puan bazında gerekli minimum fark	int
SYMBOL_TRADE_FREEZE_LEVEL	Puan bazında, alım-satım işlemlerinin dondurulmasına	int

Tanıtıcı	Açıklama	Tip
	olan uzaklık	
SYMBOL_TRADE_EXEMODE	İşlem gerçekleştirme modu	ENUM_SYMBOL_TRADE_EXECUTION
SYMBOL_SWAP_MODE	Swap hesaplama modeli	ENUM_SYMBOL_SWAP_MODE
SYMBOL_SWAP_ROLLOVER3DAYS	3 günlük swap bedelinin alındığı gün	ENUM_DAY_OF_WEEK
SYMBOL_MARGIN_HEDGED_USE_LEG	Hedgeli teminatın uzun bacak (Alış veya Satış) kullanıl	bool

Tanıtıcı	Açıklama	Tip
	arakhesaplanması	
SYMBOL_EXPIRATION_MODE	İzin verilen emir zaman-aşımı modlarının bayrakları	int
SYMBOL_FILLING_MODE	izin verilen emir uygulama modu bayrakları	int
SYMBOL_ORDER_MODE	İzin verilen emir tipi bayrakları	int
SYMBOL_ORDER_GTC_MODE	Kar Al ve	ENUM_SYMBOL_ORDER_GTC_MODE

Tanıtıcı	Açıklama	Tip
	Zarar Dur dur emirleri için zaman aşımı, SYMBOL_EXPIRATION_MODE=SYMBOL_EXPIRATION_GTC ise (İptale kadar geçerli)	
SYMBOL_OPTION_MODE	Opsiyon tipi	ENUM_SYMBOL_OPTION_MODE
SYMBOL_OPTION_RIGHT	Option right (Call/Put)	ENUM_SYMBOL_OPTION_RIGHT

SymbolInfoDouble() fonksiyonu için

ENUM_SYMBOL_INFO_DOUBLE

Tanıtıcı	Açıklama	Tip
SYMBOL_BID	Bid - en iyi satış teklifi	double
SYMBOL_BIDHIGH	Günün en yüksek Satış değeri	double
SYMBOL_BIDLOW	Günün en düşük Satış değeri	double
SYMBOL_ASK	Ask - en iyi alım teklifi	double
SYMBOL_ASKHIGH	Günün en yüksek Alış değeri	double
SYMBOL_ASKLOW	Günün en düşük Alış değeri	double
SYMBOL_LAST	Son işlemin fiyatı	double
SYMBOL_LASTHIGH	Günün en yüksek Last değeri	double
SYMBOL_LASTLOW	Günün en düşük Last değeri	double
SYMBOL_VOLUME_REAL	Son işlemin	double

Tanıtıcı	Açıklama	Tip
	Hacmi	
SYMBOL_VOLUMEHIGH_REAL	Günün en yüksek Hacmi	double
SYMBOL_VOLUMELOW_REAL	Günün en düşük Hacmi	double
SYMBOL_OPTION_STRIKE	Opsiyonun işlem fiyatı. Bir opsiyon alıcısının dayanak varlığı alabileceği (alış opsiyonu) veya satabileceği (satış opsiyonu) ve bir opsiyon satıcısının uygun miktarda dayanak varlığı almaya veya satmaya zorunlu olduğu fiyat değeri.	double

Tanıtıcı	Açıklama	Tip
SYMBOL_POINT	Sembol puan değeri	double
SYMBOL_TRADE_TICK_VALUE	SYMBOL_TRADE_TICK_VALUE_PROFIT değeri	double
SYMBOL_TRADE_TICK_VALUE_PROFIT	Karlı olabilecek bir pozisyon için hesaplanan tik fiyatı	double
SYMBOL_TRADE_TICK_VALUE_LOSS	Kaybetmekte olan bir pozisyon için hesaplanan tik fiyatı	double
SYMBOL_TRADE_TICK_SIZE	En düşük fiyat değişimi	double
SYMBOL_TRADE_CONTRACT_SIZE	Alım-satım sözleşme büyüklüğü	double
SYMBOL_TRADE_ACCRUED_INTEREST	Birikmiş faiz- birikmiş kupon faizi (kuponun basımından	double

Tanıtıcı	Açıklama	Tip
	veya son faiz ödemesinden itibaren geçen günler bazında hesaplanan kupon faizi payı)	
SYMBOL_TRADE_FACE_VALUE	Yazılı değer - basımcı tarafından ayarlanmış olan ilk bono değeri	double
SYMBOL_TRADE_LIQUIDITY_RATE	Likidite Oranı aktifin teminat olarak kullanılabilen kısmıdır.	double
SYMBOL_VOLUME_MIN	Bir işlem için gerekenen düşük hacim	double
SYMBOL_VOLUME_MAX	Bir işlemde kullanılabilecek en yüksek hacim	double

Tanıtıcı	Açıklama	Tip
SYMBOL_VOLUME_STEP	Bir işlemde kullanılabilecek en düşük hacim değişim birimi	double
SYMBOL_VOLUME_LIMIT	Bir sembold eki tek yönlü (alış veya satış) açık emirler ve bekleyen emirler için izin verilen maksimum toplam hacim. Örneğin 5 lotluk bir sınırlama ile, 5 lotluk hacimde açık bir alım pozisyonuna sahip olabilir ve yine 5 lotluk hacimde bir Sell Limit bekleye	double

Tanıtıcı	Açıklama	Tip
	n emri girebilirsiniz. Ama bu durumda (tek yöndeki toplam hacim, belirlenen sınırlamayı geçemeyeceğinden) bir Buy Limit bekleyen emri veya 5 lottan büyük bir Sell Limit emri giremezsiniz.	
SYMBOL_SWAP_LONG	Uzun swap değeri	double
SYMBOL_SWAP_SHORT	Kısa swap değeri	double
SYMBOL_SWAP_SUNDAY	Pazar gününden bir sonraki güne devreden gecelik pozisyonlar için swap hesapla	double

Tanıtıcı	Açıklama	Tip
	ma oranı (SYMBOL_SWAP_LONG veya SYMBOL_SWAP_SHORT) Şu değerler desteklenmektedir: • 0 - swap yok • 1 - tek swap • 3 - üçlü swap	
SYMBOL_SWAP_MONDAY	Pazarteside n salıya devreden	double

Tanıtıcı	Açıklama	Tip
	gecelik pozisyonlar için swap hesaplama oranı (SYMBOL_SWAP_LONG veya SYMBOL_SWAP_SHORT)	
SYMBOL_SWAP_TUESDAY	Salıdan çarşamba devreden gecelik pozisyonlar için swap hesaplama oranı (SYMBOL_SWAP_LONG veya SYMBOL_SWAP_SHORT)	double
SYMBOL_SWAP_WEDNESDAY	Çarşambadan perşembeye devreden gecelik pozisyonlar için swap hesaplama oranı (SYMBO	double

Tanıtıcı	Açıklama	Tip
	L_SWAP_LONG veya SYMBOL_SWAP_SHORT)	
SYMBOL_SWAP_THURSDAY	Perşembeden cumaya devreden gecelik pozisyonlar için swap hesaplama oranı (SYMBOL_SWAP_LONG veya SYMBOL_SWAP_SHORT)	double
SYMBOL_SWAP_FRIDAY	Cumadan cumarteesiye devreden gecelik pozisyonlar için swap hesaplama oranı (SYMBOL_SWAP_LONG veya SYMBOL_SWAP_SHORT)	double

Tanıtıcı	Açıklama	Tip
SYMBOL_SWAP_SATURDAY	Cumartesiden pazara devreden gecelik pozisyonlar için swap hesaplama oranı (SYMBOL_SWAP_LONG veya SYMBOL_SWAP_SHORT)	double
SYMBOL_MARGIN_INITIAL	Başlangıç teminatı, mevduat dövizi cinsinden bir lotluk pozisyon açmak için gerekli olan teminat miktarı anlamındadır. Bu, bir müşterinin, piyasaya girdiği zaman mal varlığının	double

Tanıtıcı	Açıklama	Tip
	kontrol edilmesi için kullanılır. SymbolInfoMarginRate() fonksiyonu, emrin tipine ve yönüne bağlı olarak, alınan marjin miktarı ile ilgili veri sağlar.	
SYMBOL_MARGIN_MAINTENANCE	Sürdürme teminatı. Eğer ayarlanmışsa, mevduat dövizi cinsinden teminat miktarını ayarlar, bir lottan itibaren değer alır. Müşterinin mal varlığını, hesap durumu değiştiğ	double

Tanıtıcı	Açıklama	Tip
	inde kontrol etmek içi kullanılır. Sürdürme teminatının 0 olması durumunda, başlangıç teminatı kullanılır. SymbolInfoMarginRate() fonksiyonu, emrin tipine ve yönüne bağlı olarak, alınan marjin miktarı ile ilgili veri sağlar.	
SYMBOL_SESSION_VOLUME	Mevcut seans pozisyonlarının özet hacim değeri	double
SYMBOL_SESSION_TURNOVER	Mevcut seans	double

Tanıtıcı	Açıklama	Tip
	cirosunun özeti	
SYMBOL_SESSION_INTEREST	Açık vadeli işlemlerin özeti	double
SYMBOL_SESSION_BUY_ORDERS_VOLUME	Alım emirlerinin mevcut hacmi	double
SYMBOL_SESSION_SELL_ORDERS_VOLUME	Satış emirlerinin mevcut hacmi	double
SYMBOL_SESSION_OPEN	Mevcut seansın açılış fiyatı	double
SYMBOL_SESSION_CLOSE	Mevcut seansın kapanış fiyatı	double
SYMBOL_SESSION_AW	Mevcut seansın ağırlıklı ortalama fiyatı	double
SYMBOL_SESSION_PRICE_SETTLEMENT	Mevcut seanstaki uzlaşma fiyatı	double
SYMBOL_SESSION_PRICE_LIMIT_MIN	Mevcut seansın en düşük fiyatı	double
SYMBOL_SESSION_PRICE_LIMIT_MAX	Mevcut seansın en	double

Tanıtıcı	Açıklama	Tip
	yüksek fiyatı	
SYMBOL_MARGIN_HEDGED	Hedge'li pozisyonun (aynı sembol üzerinde ters yönde açılmış pozisyonlar) bir lotluk hacmi için belirtilen teminat değeri veya işlem büyüklüğü. Hedge'li pozisyonlar için iki tür teminat hesaplama yöntemi bulunur. Kulllanılan hesaplama yöntemi aracı kurum tarafından belirlenir. Temel hesplama:	double

Tanıtıcı	Açıklama	Tip
	• Eğer belli bir sembol için bir başlangıç teminatı (SYMBOL_MARGIN_INITIAL) belirtilmişse, hedge'li teminat değeri de parasal cinsten kesin bir değerile belirtilir. • Başlangıç teminatı belirtilmemişse (0 ise), SYMBOL_MARGIN_HEDGED	

Tanıtıcı	Açıklama	Tip
	değeri sözleşme büyüklüğüne eşit olur. Bu, finansal enstrümanın tipine göre belirlenmiş bir formül ile teminatın hesaplanması için kullanılır (SYMBOL_TRADE_CALC_MODE). En büyük pozisyon için hesaplama: • SYMBOL_MARGIN_HEDGED değeri	

Tanıtıcı	Açıklama	Tip
	hesaba katılmaz. • Sembol üzerindeki tüm kısa ve uzun pozisyonların hacimleri hesaplanır. • Her bir yön için bir bir ağırlıklı açılış fiyatı ortalaması ve bir mevduat dövizi dönüşüm oranı hesaplanır. • Sonra, sembol tipine göre seçien	

Tanıtıcı	Açıklama	Tip
	uygun bir formülle (SYMBOL_TRADE_CALC_MODE) uzun ve kısa bölümler için teminat değeri hesaplanır. • Hesaplanan değerlerin en büyüğü teminat değeri olarak alınır.	
SYMBOL_PRICE_CHANGE	Mevcut fiyatın bir önceki işlem gününün sonuna göre % olarak	double

Tanıtıcı	Açıklama	Tip
	değişimi	
SYMBOL_PRICE_VOLATILITY	% olarak fiyat volatilitesi	double
SYMBOL_PRICE_THEORETICAL	Teorik opsiyon fiyatı	double
SYMBOL_PRICE_DELTA	Opsiyon/varant delta, dayanak varlık fiyatı 1 birim değiştiğinde opsiyon fiyatının ne kadar değiştiğini gösterir	double
SYMBOL_PRICE_THETA	Opsiyon/varant teta, opsiyon fiyatının geçici bir düşüş (diğer bir deyişle, vade bitim tarihi yaklaştığında) nedeniyle her gün kaybettiği puan	double

Tanıtıcı	Açıklama	Tip
	sayısını gösterir	
SYMBOL_PRICE_GAMMA	Opsiyon/varant gama, deltanın değişim oranını gösterir - opsiyon primi ne kadar hızlı veya yavaş değişiyor	double
SYMBOL_PRICE_VEGA	Opsiyon/varant vega, volatilite %1 değiştiğinde opsiyon fiyatının ne kadar puan değiştiğini gösterir	double
SYMBOL_PRICE_RHO	Opsiyon/varant rho, teorik opsiyon fiyatının faiz oranının %1 değişimine olan duyarlılı	double

Tanıtıcı	Açıklama	Tip
	ğını yansıtır	
SYMBOL_PRICE_OMEGA	Opsiyon/varant omega. Opsiyon esnekliği - dayanak varlığın fiyatındaki yüzde değişim için opsiyon fiyatındaki nispi yüzde değişim	double
SYMBOL_PRICE_SENSITIVITY	Opsiyon/varant hassasiyeti, opsiyonun fiyatının bir puan değişmesi için opsiyonun dayanak varlığının fiyatının kaç puan değişmesi gerektiğini gösterir	double

SymbolInfoString() fonksiyonu için

ENUM_SYMBOL_INFO_STRING

Tanıtıcı	Açıklama	Tip
SYMBOL_BASIS	Bir türev ürünün dayanak varlığı	string
SYMBOL_CATEGORY	Alım-satım sembolün ün ait olduğu sektör veya kategorini n adı	string
SYMBOL_COUNTRY	Finansal sembolün ait olduğu ülke	string
SYMBOL_SECTOR_NAME	Finansal sembolün ait olduğu ekonomi sektörü	string
SYMBOL_INDUSTRY_NAME	Finansal sembolün ait olduğu sanayi veya sanayi dalı	string
SYMBOL_CURRENCY_BASE	Sembolün baz dövizi	string
SYMBOL_CURRENCY_PROFIT	Kar dövizi	string
SYMBOL_CURRENCY_MARGIN	Teminat döviz birimi	string
SYMBOL_BANK	Mevcut fiyat teklifinin dağıtıcısı	string
SYMBOL_DESCRIPTION	Sembol açıklamas ı	string

Tanıtıcı	Açıklama	Tip
SYMBOL_EXCHANGE	Finansal sembolün alım-satım yapıldığı borsanın adı	string
SYMBOL_FORMULA	Formül, kullanıcı tanımlı sembolün fiyatlandırılması için kullanılır. Formülde kullanılan bir finansal sembolün adı bir rakamla başlıyorsa veya özel bir karakter (">" "", ".", "-", "&", "#" vb.) içeriyorsa, bu sembol adının çevresinde tırnak işaretleri kullanılmalıdır. • Sentetik sym	string

Tanıtıcı	Açıklama	Tip
	bol:"@ESU19"/EURCAD • Takvimaralığı:"Si-9.13"-"Si-6.	

Tanıtıcı	Açıklama	Tip
	13" • Euroendeksi:34.38805726*pow(EURUSD,0.31555)*pow	

Tanıtıcı	Açıklama	Tip
	(EURGBP,0.30560)*pow(EURJPY,0.18911)*pow(EURCHF,0.1	

Tanıtıcı	Açıklama	Tip
	113)*pow(EURSEK,0.07855)	
SYMBOL_ISIN	Bir sembolün ISIN (International Securities Identification Number) sistemindeki ismi. Uluslararası Menkul Değerleri Tanımlama Numarası, bir menkul değeri benzersiz şekilde tanımlayan 12-haneli bir	string

Tanıtıcı	Açıklama	Tip
	alfa-sayısal koddur. Bu sembol özelliğinin varlığı, alım-satım sunucusu tarafında belirlenir.	
SYMBOL_PAGE	Sembol bilgisini içeren web sayfasının adresi. Bu adres, terminalde sembol özellikleri kapsamında bir bağlantı şeklinde gösterilir	string
SYMBOL_PATH	Sembol ağacındaki veri yolu	string

Sembollerin fiyat çizelgeleri Satış veya Son işlem fiyatı temelinde çizilir. Sembol çizelgesi için seçilen fiyat tipi ayrıca terminalde çubukların nasıl oluşturulup görüntüleneceğini de etkiler. SYMBOL_CHART_MODE özelliğinin olası değerleri ENUM_SYMBOL_CHART_MODE sayımında açıklanmıştır

ENUM_SYMBOL_CHART_MODE

Tanıtıcı	Açıklama
SYMBOL_CHART_MODE_BID	Satış fiyatına göre çubuklar
SYMBOL_CHART_MODE_LAST	Son işlem fiyatına göre çubuklar

Her bir sembolde, bekleyen emirler için birkaç zaman-aşımı modu belirtilebilir. Her moda bir bayrak verilir. Bayraklar, mantıksal **VEYA** (|) işlemi kullanılarak birleştirilebilir, örneğin, SYMBOL_EXPIRATION_GTC|SYMBOL_EXPIRATION_SPECIFIED. Belli bir modun sembol için kullanımına

izin verilip verilmediğini kontrol etmek için, mantıksal **VE** (&) işleminin sonucu, bayrak modu ile karşılaştırılmalıdır.

Eğer bir sembol için SYMBOL_EXPIRATION_SPECIFIED bayrağı belirtilmişse, o zaman bir bekleyen emir gönderirken, bu emrin ne zamana kadar geçerli olacağını belirtmelisiniz.

Tanıtıcı	Değer	Açıklama
SYMBOL_EXPIRATION_GTC	1	Açık bir şekilde iptal edilmediği sürece, emir sınırsız bir zaman aralığı için geçerlidir.
SYMBOL_EXPIRATION_DAY	2	Emir gün sonuna kadar geçerlidir
SYMBOL_EXPIRATION_SPECIFIED	4	Zaman-aşımı zamanı emirde belirtilir
SYMBOL_EXPIRATION_SPECIFIED_DAY	8	Zaman-aşımı tarihi emirde belirtilir

Örnek:

```
//+------------------------------------------------------------------+
//| Belirlenen zaman-aşımı moduna izin veriliyor mu kontrol eder     |
//+------------------------------------------------------------------+
bool IsExpirationTypeAllowed(string symbol,int exp_type)
  {
//--- İzin verilen zaman-aşımı modunu tanımlayan özellik değerini al
   int expiration=(int)SymbolInfoInteger(symbol,SYMBOL_EXPIRATION_MODE);
//--- exp_type moduna izin verilmişse true dönüşü yap
   return((expiration&exp_type)==exp_type);
  }
```

SYMBOL_EXPIRATION_MODE özelliği SYMBOL_EXPIRATION_GTC (iptale kadar geçerli) olarak ayarlanmışsa, bekleyen emirler ve durdurma emirleri (Kar Al, Zarar Durdur) de ENUM_SYMBOL_ORDER_GTC_MODE sayımına göre ayarlanmalıdır.

ENUM_SYMBOL_ORDER_GTC_MODE

Tanıtıcı	Açıklama
SYMBOL_ORDERS_GTC	Bekleyen emirler ve durdurma emirleri açık şekilde iptal edilene kadar sınırsız süre için geçerlidir
SYMBOL_ORDERS_DAILY	Emirler sadece ilgili işlem günü için geçerlidir. Gün sonunda tüm durdurma emirleri ve bekleyen emirler silinir.
SYMBOL_ORDERS_DAILY_EXCLUDING_STOPS	İşlem günü değiştiğinde sadece bekleyen emirler silinir durdurma emirleri (Kar Al, Zarar Durdur) korunur.

Emir gönderirken emirde belirlenen hacmin yerine getirme politikasını belirtebiliriz. Her bir sembol için mevcut hacme dayalı emir gerçekleştirme seçenekleri tabloda belirtilmiştir. Bayrak kombinasyonu aracılığıyla her bir enstrüman için birkaç mod ayarlamak mümkündür. Bayrak kombinasyonu, mantıksal **OR** (|) operatörüyle ifade edilir, örneğin SYMBOL_FILLING_FOK|SYMBOL_FILLING_IOC. Bir enstrüman için belirli bir moda izin verilip verilmediğini kontrol etmek için mantıksal **AND** (&) sonucunu modun bayrağı ile karşılaştırın - örnek.

Hacim yerine getirme politikası	ID	Değer	Açıklama
Kalanı İptal Et (Fill or Kill, FOK)	SYMBOL_FILLING_FOK	1	Bir emir yalnızca belirtilen hacim değerçek

Hacim yerine getirme politikası	ID	Değer	Açıklama
			leştirilebilir. Finanslenstrümanı için piyasada g

Hacim yerine getirme politikası	ID	Değer	Açıklama
			erekli miktar daha hacim mevcut değilse, emir gerçekle

Hacim yerine getirme politikası	ID	Değer	Açıklama
			şmez. İstenen hacim mevcut bir kaçfiyatı içerebilir.

Hacim yerine getirme politikası	ID	Değer	Açıklama
			Emir gönderirken bu politika için ORDER_FILLING_FOK

Hacim yerine getirme politikası	ID	Değer	Açıklama
			yerine getirme tipi belirtilmelidir. FOK emirlerini kullanır

Hacim yerine getirme politikası	ID	Değer	Açıklama
			İşlanmaimkanıticaretsunucususutarafındanbelirlir

Hacim yerine getirme politikası	ID	Değer	Açıklama
			lenir.
Anında ya da İptal Et (Immediate or Cancel, IOC)	SYMBOL_FILLING_IOC	2	Yatırımcı, emirde belirtilen dahil indepiyasada

Hacim yerine getirme politikası	ID	Değer	Açıklama
			bulunan maksimum hacim değişlemgerçekleştirmeyi k

Hacim yerine getirme politikası	ID	Değer	Açıklama
			abuledeer.Talebiintamolarakkkarşılanamamasıdur

Hacim yerine getirme politikası	ID	Değer	Açıklama
			umunda, emir varolan hacim değer gerçekleştirilecek ve

Hacim yerine getirme politikası	ID	Değer	Açıklama
			kalan hacim iptal edilecektir. Emir gönderirken bu

Hacim yerine getirme politikası	ID	Değer	Açıklama
			olitikaiçinORDER_FILLING_IOCyerinegetirmetipibe

Hacim yerine getirme politikası	ID	Değer	Açıklama
			lirtilmelidir. IOC emirlerinin kullanmaimkanıticaretes

Hacim yerine getirme politikası	ID	Değer	Açıklama
			unucusutarafındanbelirlenir.
Pasife yerleştir	SYMBOL_FILLING_BOC	4	BOC(BookOrCancel)p

Hacim yerine getirme politikası	ID	Değer	Açıklama
			olitikası, emrin yalnızca Piyasa Derinliğine ye

Hacim yerine getirme politikası	ID	Değer	Açıklama
			rleştirilebileceğini ve hemen gerçekleştirile

Hacim yerine getirme politikası	ID	Değer	Açıklama
			meyeceğini belirtir. Emir yerleştirildiğinda

Hacim yerine getirme politikası	ID	Değer	Açıklama
			hemen yerine getirilebilir ise bu emir iptal edi

Hacim yerine getirme politikası	ID	Değer	Açıklama
			lir. Aslında buyerine getirme politikası, yerl

Hacim yerine getirme politikası	ID	Değer	Açıklama
			eştirilenemrinfiyatınınmevcutpiyasadadaha

Hacim yerine getirme politikası	ID	Değer	Açıklama
			kötü olacağının garantiedeer. BOC emirlerin pasift

Hacim yerine getirme politikası	ID	Değer	Açıklama
			işaret yapmak için kullanılır: emrin yerleştiri

Hacim yerine getirme politikası	ID	Değer	Açıklama
			ldiği anda yerine getirilmeyeceği ve böylece mev

Hacim yerine getirme politikası	ID	Değer	Açıklama
			cutlikiditeyietkilememeyeceğigarantiedilir.

Hacim yerine getirme politikası	ID	Değer	Açıklama
			Yalnızca limit ve stop limit emirlerinde desteklenmiyor

Hacim yerine getirme politikası	ID	Değer	Açıklama
			ektedir, yani SYMBOL_ORDER_MODE bayrağı SYMBOL

Hacim yerine getirme politikası	ID	Değer	Açıklama
			_ORDER_LIMIT ve/veya SYMBOL_ORDER_STOP_LIMIT

Hacim yerine getirme politikası	ID	Değer	Açıklama
			değerini içermelidir.
Dönüş	Tanımlayıcı yok		Hacmin kısmi olarak yerine

Hacim yerine getirme politikası	ID	Değer	Açıklama
			getirilmesi durumunda, hacmi artırmış piyasa ve yalim

Hacim yerine getirme politikası	ID	Değer	Açıklama
			itemri iptal edilmez, daha sonra için işlenir. Emirg

Hacim yerine getirme politikası	ID	Değer	Açıklama
			önderirkenbupolitikaiçinORDER_FILLING_RETURNy

Hacim yerine getirme politikası	ID	Değer	Açıklama
			erine getirme tipi belirtilmelidir. Piyasa Fiyatından

Hacim yerine getirme politikası	ID	Değer	Açıklama
			İşlemGerçekleştirmemodundadönüşemirlerinei

Hacim yerine getirme politikası	ID	Değer	Açıklama
			zinverilmez (piyasa sağlayıcısından gerçekleş

Hacim yerine getirme politikası	ID	Değer	Açıklama
			tirme_SYMBOL_TRADE_EXECUTION_MARKET).

OrderSend() fonksiyonunu kullanarak bir ticaret talebi gönderirken, *type_filling* alanında, yani özel MqlTradeRequest yapısında gerekli hacim yerine getirme politikası ayarlanabilir. ENUM_ORDER_TYPE_FILLING sayımındaki değerler kullanılabilir. Yerine getirme tipi belirtilmezse, ticaret talebinde ORDER_FILLING_RETURN otomatik olarak ayarlanır. ORDER_FILLING_RETURN yerine

getirme tipi, "Piyasa fiyatından işlem gerçekleştirme" (SYMBOL_TRADE_EXECUTION_MARKET) dışında her işlem gerçekleştirme modunda etkinleştirilir.

Geçerli zamanda (yürürlükteki zaman) işlem gerçekleştirmek için bir işlem talebi gönderirken, finansal piyasaların, talep edilen hacmin tamamının belirli bir finansal enstrüman için istenen fiyatta mevcut olduğuna dair hiçbir garanti vermediğini unutmayın. Bu nedenle, gerçek zamanlı ticaret işlemleri, fiyatla ve hacimle ilişkili işlem gerçekleştirme modları kullanılarak düzenlenir. Modlar veya işlem gerçekleştirme politikaları, fiyatın değiştiği veya istenen hacmin **o anda** tam olarak karşılanamadığı durumlar için kuralları tanımlar.

İşlem gerçekleştirme modu	Açıklama	ENUM_SYMBOL_TRADE_EXECUTION'daki değer
İşlem gerçekleştirme modu (İşlem Gerçekleştirme Talebi)	Brokerdan alınan fiyattan bir piyasa emri gerçekleştirme. Belirli bir piyasa emrinin fiyatları, emir gönderilmeden önce brokerdan talep edilir. Fiyatlar alındıktan sonra, elde edilen fiyattan emrin gerçekleştirilmesi onaylanabilir	SYMBOL_TRADE_EXECUTION_REQUEST

İşlem gerçekleştirme modu	Açıklama	ENUM_SYMBOL_TRADE_EXECUTION'daki değer
	veya reddedilebilir.	
Anında İşlem Gerçekleştirme (Anında İşlem Gerçekleştirme)	Belirtilen fiyattan anında bir piyasa emri gerçekleştirme. Gerçekleştirilmek üzere bir işlem talebi gönderirken, platform otomatik olarak mevcut fiyatları emre ekler. • Broker fiyatıka	SYMBOL_TRADE_EXECUTION_INSTANT

İşlem gerçekleştirme modu	Açıklama	ENUM_SYMBOL_TRADE_EXECUTION'daki değer
	bulederse, emir gerçekleştirilir. • Brokerfiyatıkabul	

İşlem gerçekleştirme modu	Açıklama	ENUM_SYMBOL_TRADE_EXECUTION'daki değer
	etmezse, bir "Requote" gönderilir - broker bu emri ingerç	

İşlem gerçekleştirme modu	Açıklama	ENUM_SYMBOL_TRADE_EXECUTION'daki değer
	eklestirilebilecegi fiyatlari geri dondurur.	
Piyasa Fiyatından İşlem Gerçekleştirme (Piyasa Fiyatından İşlem Gerçekleştirme)	Broker yatırımcıyla herhangi bir ek	SYMBOL_TRADE_EXECUTION_MARKET

İşlem gerçekleştirme modu	Açıklama	ENUM_SYMBOL_TRADE_EXECUTION'daki değer
	iletişim yapmadan emir gerçekleştirme fiyatı hakkında karar verir. Emrin bu modda gönderilmesi, o anki fiyattan emrin gerçekleşmesine öncedeN izin verilmesi anlamına gelir.	
Exchange İşlem Gerçekleştirme (Exchange İşlem Gerçekleştirme)	Ticaret işlemleri mevcut piyasa fiyatları üzerinden gerçekleştirilir.	SYMBOL_TRADE_EXECUTION_EXCHANGE

Geçerli işlem gerçekleştirme zamanıyla bir emir göndermeden önce, ORDER_TYPE_FILLING değerinin (hacimle ilişkili işlem gerçekleştirme tipi) doğru ayarlanması için, SYMBOL_FILLING_MODE özellik değerini elde etmek adına her bir finansal enstrümanla SymbolInfoInteger() fonksiyonunu (bir bayrak kombinasyonu şeklinde sembol için izin verilen hacimle ilişkili işlem gerçekleştirme tiplerini gösterir)

kullanabilirsiniz. ORDER_FILLING_RETURN tipi, "Piyasa fiyatından işlem gerçekleştirme" modu (SYMBOL_TRADE_EXECUTION_MARKET) dışında her zaman etkindir.

İşlem gerçekleştirme moduna bağlı olarak hacim yerine getirme tiplerinin kullanımı aşağıdaki tablo halinde gösterilebilir:

İşlem Gerçekleştirme Tipi\Hacim Yerine Getirme Politikası	Kalanı İptal Et (FOK ORDER_FILLING_FOK)	Anında ya da İptal Et (IOC ORDER_FILLING_IOC)	Dönüş (Return ORDER_FILLING_RETURN)
Anında İşlem Gerçekleştirme (SYMBOL_TRADE_EXECUTION_INSTANT)	+ (sembol ayarından bağımsız olarak)	+ (sembol ayarından bağımsız olarak)	+ (her zaman)
İşlem Gerçekleştirme Talebi SYMBOL_TRADE_EXECUTION_REQUEST	+ (sembol ayarından bağımsız olarak)	+ (sembol ayarından bağımsız olarak)	+ (her zaman)
Piyasa Fiyatından İşlem Gerçekleştirme SYMBOL_TRADE_EXECUTION_MARKET	+ (sembol ayarlarından ayarlanır)	+ (sembol ayarlarından ayarlanır)	- (sembol ayarlarından bağımsız olarak devre dışı)
Exchange İşlem Gerçekleştirme SYMBOL_TRADE_EXECUTION_EXCHANGE	+ (sembol ayarlarından ayarlanır)	+ (sembol ayarlarından ayarlanır)	+ (her zaman)

Bekleyen emirler olması durumunda, bu tür emirler gönderme sırasında gerçekleştirilmeleri üzere tasarlanmadığından, işlem gerçekleştirme tipinden (SYMBOL_TRADE_EXEMODE) bağımsız olarak ORDER_FILLING_RETURN hacim yerine getirme tipi kullanılmalıdır. Bekleyen emirleri kullanırken, yatırımcı, emirdeki koşulların işlem için karşılandığında, brokerın borsa tarafından desteklenen hacim yerine getirme tipini kullanacağını peşinen kabul eder.

Örnek:

```
//+------------------------------------------------------------------+
//| Belirtilen emir türüne izin veriliyor mu kontrol eder            |
//+------------------------------------------------------------------+
bool IsFillingTypeAllowed(string symbol,int fill_type)
  {
//--- yerine getirme modunu açıklayan özelliğin değerini elde et
   int filling=(int)SymbolInfoInteger(symbol,SYMBOL_FILLING_MODE);
//--- fill_type moduna izin veriliyorsa 'true' geri döndür
   return((filling&fill_type)==fill_type);
  }
```

OrderSend() fonksiyonu kullanarak bir alım-satım isteği gönderirken, bazı işlemler için ENUM_ORDER_TYPE sayımından bir emir tipi belirtilmelidir. Belirli bir sembol için tüm emir tiplerine izin verilmeyebilir. SYMBOL_ORDER_MODE özelliği, izin verilen emir tiplerinin bayraklarını tarif eder.

Tanıtıcı	Değer	Açıklama
SYMBOL_ORDER_MARKET	1	Piyasa emirlerine (Alış ve Satış) izin verilir
SYMBOL_ORDER_LIMIT	2	Limitli emirlere izin verilir (Buy Limit ve Sell Limit)
SYMBOL_ORDER_STOP	4	Stop (durdurma) emirlerine izin verilir (Buy Stop ve Sell Stop)
SYMBOL_ORDER_STOP_LIMIT	8	Stop-limit emirlerine izin verilir (Buy Stop Limit ve Sell Stop Limit)
SYMBOL_ORDER_SL	16	Zarar Durdur emrine izin verilir
SYMBOL_ORDER_TP	32	Kar Al emrine izin verilir
SYMBOL_ORDER_CLOSEBY	64	Bir 'Pozisyon ile Kapa' (bir pozisyonu aynı enstrüman değerine sahip ters yönlü bir açık pozisyonla kapama) işlemi için izin bayrağı. Bu özellik hedge muhasebeli (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) hesaplar için ayarlanır

Örnek:

```
//+------------------------------------------------------------------+
//| Fonksiyon, bir sembol için izin verilen emir tiplerini çıktılar  |
//+------------------------------------------------------------------+
void Check_SYMBOL_ORDER_MODE(string symbol)
  {
//--- izin verilen emir tiplerini açıklayan özelliğin değerini al
   int symbol_order_mode=(int)SymbolInfoInteger(symbol,SYMBOL_ORDER_MODE);
//--- piyasa emirleri (Piyasa İşlemleri) için kontrol et
   if((SYMBOL_ORDER_MARKET&symbol_order_mode)==SYMBOL_ORDER_MARKET)
      Print(symbol+": Piyasa emirlerine (Alış ve Satış) izin veriliyor");
//--- Limit emirleri için kontrol et
   if((SYMBOL_ORDER_LIMIT&symbol_order_mode)==SYMBOL_ORDER_LIMIT)
      Print(symbol+": Limit Alış ve Limit Satış emirlerine izin veriliyor");
//--- Stop emirleri için kontrol et
   if((SYMBOL_ORDER_STOP&symbol_order_mode)==SYMBOL_ORDER_STOP)
      Print(symbol+": Stop Alış ve Stop Satış emirlerine izin veriliyor");
//--- Stop Limit emirleri için kontrol et
   if((SYMBOL_ORDER_STOP_LIMIT&symbol_order_mode)==SYMBOL_ORDER_STOP_LIMIT)
      Print(symbol+": Stop Limit Alış ve Stop Limit Satış emirlerine izin veriliyor");
//--- Zarar Durdur emrine izin veriliyor mu kontrol et
   if((SYMBOL_ORDER_SL&symbol_order_mode)==SYMBOL_ORDER_SL)
```

```
      Print(symbol+": Zarar Durdur emirlerine izin veriliyor");
//--- Kar Al emrine izin veriliyor mu kontrol et
   if((SYMBOL_ORDER_TP&symbol_order_mode)==SYMBOL_ORDER_TP)
      Print(symbol+": Kar Al emirlerine izin veriyor");
//--- pozisyonun zıt olanla kapatılmasına izin verilip verilmediğini kontrol et
   if((SYMBOL_ORDER_TP&symbol_order_mode)==SYMBOL_ORDER_CLOSEBY)
      Print(symbol+": Close by işlemlerine izin verilmektedir");
//---
  }
```

ENUM_SYMBOL_CALC_MODE sayımı, bir sembol için teminat gereksinimlerinin nasıl hesaplandığı hakkında bilgi edinmek için kullanılır.

ENUM_SYMBOL_CALC_MODE

Tanıtıcı	Açıklama	Formül
SYMBOL_CALC_MODE_FOREX	Forex modu - Forex için kar ve teminat değerlerinin hesaplanması	Teminat: Lot * SözleşmeBüyüklüğü / Kaldıraç * Margin_Rate Kar: (KapanışFiyatı - AçılışFiyatı) * SözleşmeBüyüklüğü * Lot
SYMBOL_CALC_MODE_FOREX_NO_LEVERAGE	Forex Kaldıraçsız modu - kaldıracı hesaba katmadan Fore	Margin: Lots * Contract_Size * Margin_Rate Profit: (close_price - open_price) * Contract_Size * Lots

Tanıtıcı	Açıklama	Formül
	x sembolleri için kar ve teminat hesaplama	
SYMBOL_CALC_MODE_FUTURES	Vadeli mod u - vadeli ürünler için kar ve teminat değerlerin in hesaplan ması	Teminat: Lot * BaşlangıçTeminatı * Margin_Rate Kar: (KapanışFiyatı - AçılışFiyatı) * TikFiyatı / TikBüyüklüğü * Lot * Margin_Rate
SYMBOL_CALC_MODE_CFD	CFD mod u - CFD için kar ve teminat değerlerin in hesaplan ması	Teminat: Lot * SözleşmeBüyüklüğü * PiyasaFiyatı * Margin_Rate Kar: (KapanışFiyatı - AçılışFiyatı) * SözleşmeBüyüklüğü * Lot

Tanıtıcı	Açıklama	Formül
SYMBOL_CALC_MODE_CFDINDEX	CFD indeks modu - CFD için kar ve teminat değerlerinin endekslerle hesaplanması	Teminat: (Lot * SözleşmeBüyüklüğü * PiyasaFiyatı) * TikFiyatı / TikBüyüklüğü * Margin_Rate Kar: (KapanışFiyatı - AçılışFiyatı) * SözleşmeBüyüklüğü * Lot
SYMBOL_CALC_MODE_CFDLEVERAGE	Kaldıraçlı CFD modu - CFD üzerinde kaldıraçlı alım-satım için için kar ve teminat değerlerinin hesaplanması	Teminat: (Lot * SözleşmeBüyüklüğü * PiyasaFiyatı) / Kaldıraç * Margin_Rate Kar: (KapanışFiyatı - AçılışFiyatı) * SözleşmeBüyüklüğü * Lot
SYMBOL_CALC_MODE_EXCH_STOCKS	Borsa	Teminat: Lot * SözleşmeBüyüklüğü * LastFiyatı * Margin_Rate

Tanıtıcı	Açıklama	Formül
	modu - hisse senedi piyasalarında, menkul değer alım-satımı için kar ve teminat değerlerinin hesaplanması	Kar: (KapanışFiyatı - AçılışFiyatı) * SözleşmeBüyüklüğü * Lot
SYMBOL_CALC_MODE_EXCH_FUTURES	Vadeli modu - hisse senedi piyasasında vadeli işlem sözleşmelerinin alım-satımı için kar	Teminat: Lot * BaşlangıçTeminatı * Margin_Rate veya Lot * SürdürmeTeminatı * Margin_Rate Kar: (KapanışFiyatı - AçılışFiyatı) * Lot * TikFiyatı / TikBüyüklüğü

Tanıtıcı	Açıklama	Formül
	ve teminat değerlerinin hesaplanması	
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	FORTS Vadeli mod u - FORTS üzerinde vadeli işlem sözleşmelerinin alım-satımı için kar ve teminat değerlerinin hesaplanması. Teminat, Teminatin dirimi (isko	Teminat: Lot * BaşlangıçTeminatı * Margin_Rate veya Lot * SürdürmeTeminatı * Margin_Rate Kar: (KapanışFiyatı - AçılışFiyatı) * Lot * TikFiyatı / TikBüyüklüğü

Tanıtıcı	Açıklama	Formül
	nto) sapmasının miktarı kadar, şu kurallara göre azaltılabilir: 1. Bir uzun pozisyonun (alış emri) fiyatı tahmin edilen fiyattan az ise, TeminatİndirimiLot*((AnlaşmaFiyatı - EmirFiyatı) * TikFiyatı / TikB	

Tanıtıcı	Açıklama	Formül
	üyüklüğü) 2. Bir kısa pozisyonun (satış emri) fiyatı tahmin edilen fiyatı aşıyorsa, TeminatİndirimiLot*((EmirFiyatı - AnlaşmaFiyatı) * TikFiyatı / TikBüyüklüğü) burada: ○ Anlaşmaf	

Tanıtıcı	Açıklama	Formül
	iyatı - birönceki seans ıntahmine dilenfiyatı; ○ Emir Fiy	

Tanıtıcı	Açıklama	Formül
	atı-emir (istek) için de ayarlanan açıklış fiyatı ve yapo	

Tanıtıcı	Açıklama	Formül
	zisyonfiyatının anağırlıklıortalaması; ○ TikFiyatı - bir	

Tanıtıcı	Açıklama	Formül
	puanlık fiyat değişiminin minimal yetiTik Büyüklüğü - mini	

Tanıtıcı	Açıklama	Formül
	mumfiyatdeğişimbirimi	
SYMBOL_CALC_MODE_EXCH_BONDS	Borsa Tahviller modu - Borsadaki alım-satım tahvilleri için teminat ve karın hesaplaması	Margin: Lots * ContractSize * FaceValue * open_price * /100 Profit: Lots * close_price * FaceValue * Contract_Size + AccruedInterest * Lots * ContractSize
SYMBOL_CALC_MODE_EXCH_STOCKS_MOEX	Borsa MOE	Margin: Lots * ContractSize * LastPrice * Margin_Rate

Tanıtıcı	Açıklama	Formül
	X Hisse Senedi modu - MOEX'te alım-satım menkul değerleri için teminat ve karın hesaplanması	Profit: (close_price - open_price) * Contract_Size * Lots
SYMBOL_CALC_MODE_EXCH_BONDS_MOEX	Borsa MOEX Tahviller modu - MOEX'te alım-satım tahvilleri için teminat ve karın hesaplanması	Margin: Lots * ContractSize * FaceValue * open_price * /100 Profit: Lots * close_price * FaceValue * Contract_Size + AccruedInterest * Lots * ContractSize

Tanıtıcı	Açıklama	Formül
SYMBOL_CALC_MODE_SERV_COLLATERAL	Kolateral mod - (teminat modu) alım-satım hesabı üzerindeki bir sembol alınıp satılamayan bir varlık - yani teminat- olarak kullanılır. Açık pozisyonların piyasa değeri hacme, mevcut piyasa fiyatı	Teminat: yok Kar: yok Piyasa Değeri: Lot * SözleşmeBüyüklüğü * PiyasaFiyatı * LikiditeOranı

Tanıtıcı	Açıklama	Formül
	na, sözleşme büyüklüğüne ve likidite oranına göre hesaplanır. Değer, Aktiflere dolayısıyla varlığa eklenir. Bu tip semboller ile açılan pozisyonlar serbest teminat miktarını artırır ve alınıp-satılabilen enstr	

Tanıtıcı	Açıklama	Formül
	üma nlar için ek temi nat olara k kulla nılır.	

Semboller için birkaç alım-satım modu bulunmaktadır. Belirli bir sembolün alım-satım modları hakkındaki bilgi, ENUM_SYMBOL_TRADE_MODE sayımının değerlerine yansır.

ENUM_SYMBOL_TRADE_MODE

Tanıtıcı	Açıklama
SYMBOL_TRADE_MODE_DISABLED	Sembol için alım-satım devre dışı bırakılmıştır
SYMBOL_TRADE_MODE_LONGONLY	Sadece uzun pozisyonlara izin verilir
SYMBOL_TRADE_MODE_SHORTONLY	Sadece kısa pozisyonlara izin verilir
SYMBOL_TRADE_MODE_CLOSEONLY	Sadece pozisyon kapama işlemlerine izin verilir
SYMBOL_TRADE_MODE_FULL	Alım-satım kısıtlaması yoktur

Belirli bir sembol için muhtemel işlem uygulama modları, ENUM_SYMBOL_TRADE_EXECUTION sayımında tanımlanmıştır.

ENUM_SYMBOL_TRADE_EXECUTION

Tanıtıcı	Açıklama
SYMBOL_TRADE_EXECUTION_REQUEST	İstek ile uygulama
SYMBOL_TRADE_EXECUTION_INSTANT	Anında uygulama
SYMBOL_TRADE_EXECUTION_MARKET	Piyasa Uygulaması
SYMBOL_TRADE_EXECUTION_EXCHANGE	Borsa uygulaması

Pozisyon aktarımı sırasındaki swap hesabı yöntemleri, ENUM_SYMBOL_SWAP_MODE sayımında belirtilir. Swap hesaplama yöntemi, SYMBOL_SWAP_LONG ve SYMBOL_SWAP_SHORT parametrelerinin ölçüm birimlerini belirler. Örneğin, swap bedelleri müşterinin teminat döviz birimi üzerinden uygulanıyorsa, bu parametrelerin değerleri teminat döviz birimi cinsinden belirlenecektir.

ENUM_SYMBOL_SWAP_MODE

Tanıtıcı	Açıklama
SYMBOL_SWAP_MODE_DISABLED	Swap devre dışı (swap yok)
SYMBOL_SWAP_MODE_POINTS	Swap bedelleri puan bazında uygulanır
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	Swap bedelleri semboldeki baz döviz birimi üzerinden uygulanır
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	Swap bedelleri semboldeki karşıt döviz üzerinden uygulanır
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	Swap bedelleri, müşterinin teminat para birimi üzerinden uygulanır
SYMBOL_SWAP_MODE_INTEREST_CURRENT	Swap bedelleri, hesaplanma sırasındaki finansal enstrüman fiyatının belirlenen yıllık faizi üzerinden uygulanır (standart banka yılı 360 gündür)
SYMBOL_SWAP_MODE_INTEREST_OPEN	Swap bedelleri, pozisyonun açılış fiyatının belirlenen yıllık faizi üzerinden uygulanır (standart banka yılı 360 gündür)
SYMBOL_SWAP_MODE_REOPEN_CURRENT	Swap bedelleri yeniden açılan pozisyonlar üzerinden uygulanır. Alım-satım gününün sonunda pozisyon kapatılır. Ertesi gün pozisyonlar, kapanış fiyatı +/- belirlenen puan sayısı (SYMBOL_SWAP_LONG ve SYMBOL_SWAP_SHORT parametreleri) ile yeniden açılır
SYMBOL_SWAP_MODE_REOPEN_BID	Swap bedelleri yeniden açılan pozisyonlar üzerinden uygulanır. Alım-satım gününün sonunda pozisyon kapatılır. Ertesi gün, mevcut Satış fiyatı +/- belirlenen puan sayısı (SYMBOL_SWAP_LONG ve SYMBOL_SWAP_SHORT parametreleri) ile yeniden açılır

ENUM_DAY_OF_WEEK sayımı haftanın günlerini belirtmek için kullanılır.

ENUM_DAY_OF_WEEK

Tanıtıcı	Açıklama
SUNDAY	Pazar
MONDAY	Pazartesi
TUESDAY	Salı
WEDNESDAY	Çarşamba
THURSDAY	Perşembe

Tanıtıcı	Açıklama
FRIDAY	Cuma
SATURDAY	Cumartesi

Opsiyon, bir dayanak varlığı belirtilen fiyattan veya belirtilen tarihten önce alma veya satma hakkı veren (zorunlu değil) bir sözleşmedir. Şu sayımlar –opsiyon tipi ve sözleşmeye konu olan hak da dahil olmak üzere– opsiyon özelliklerini tanımlar.

ENUM_SYMBOL_OPTION_RIGHT

Tanıtıcı	Açıklama
SYMBOL_OPTION_RIGHT_CALL	Alış opsiyonu, size bir varlığı belirtilen fiyattan alma hakkı verir
SYMBOL_OPTION_RIGHT_PUT	Satış opsiyonu, size bir varlığı belirtilen fiyattan Satma hakkı verir

ENUM_SYMBOL_OPTION_MODE

Tanıtıcı	Açıklama
SYMBOL_OPTION_MODE_EUROPEAN	Avrupa opsiyonu sadece belli bir tarihte kullanılır (zaman-aşımı tarihi, işlem tarihi, dağıtım tarihi)
SYMBOL_OPTION_MODE_AMERICAN	Amerikan opsiyonu zaman aşımından önce herhangi bir işlem gününde kullanılabilir. Bunun için alıcının opsiyonu kullanabileceği bir periyot belirtilir.

Finansal enstrümanlar ekonomi sektörlerine göre sınıflandırılır. Bir ekonomi sektörü, ekonomik faaliyetin bir bölümüdür ve ekonominin diğer bölümlerinden ayrılmasına olanak sağlayan spesifik özelliklere, ekonomik hedeflere, işlevlere ve davranışlara sahiptir. ENUM_SYMBOL_SECTOR, bir ticaret enstrümanının ait olduğu ekonomi sektörlerini listeler.

ENUM_SYMBOL_SECTOR

Tanımlayıcı	Açıklama
SECTOR_UNDEFINED	Belirsiz
SECTOR_BASIC_MATERIALS	Ham maddeler
SECTOR_COMMUNICATION_SERVICES	İletişim servisleri
SECTOR_CONSUMER_CYCLICAL	Tüketici - Konjonktürel
SECTOR_CONSUMER_DEFENSIVE	Tüketici - İstikrarlı

Tanımlayıcı	Açıklama
SECTOR_CURRENCY	Para birimleri
SECTOR_CURRENCY_CRYPTO	Kripto para
SECTOR_ENERGY	Enerji
SECTOR_FINANCIAL	Finans
SECTOR_HEALTHCARE	Sağlık hizmeti
SECTOR_INDUSTRIALS	Sanayi
SECTOR_REAL_ESTATE	Gayrimenkul
SECTOR_TECHNOLOGY	Teknoloji
SECTOR_UTILITIES	Kamu hizmetleri

Her bir finansal enstrüman, belirli bir sanayi tipine veya ekonomi dalına atanabilir. Bir sanayi, ekonominin bir dalıdır ve yakından ilişkili ham maddeler, mallar veya hizmetler üretir. ENUM_SYMBOL_INDUSTRY, bir ticaret enstrümanının ait olduğu sanayileri listeler.

ENUM_SYMBOL_INDUSTRY

Tanımlayıcı	Değer	Açıklama
INDUSTRY_UNDEFINED	0	Belirsiz
Ham maddeler		
INDUSTRY_AGRICULTURAL_INPUTS	1	Tarım girdileri
INDUSTRY_ALUMINIUM	2	Alüminyum
INDUSTRY_BUILDING_MATERIALS	3	Yapı malzemeleri
INDUSTRY_CHEMICALS	4	Kimyasallar
INDUSTRY_COKING_COAL	5	Kok kömürü
INDUSTRY_COPPER	6	Bakır
INDUSTRY_GOLD	7	Altın
INDUSTRY_LUMBER_WOOD	8	Kereste ve odun üretimi
INDUSTRY_INDUSTRIAL_METALS	9	Diğer endüstriyel metaller ve madencilik
INDUSTRY_PRECIOUS_METALS	10	Diğer değerli metaller ve madencilik
INDUSTRY_PAPER	11	Kağıt ve kağıt ürünleri

Tanımlayıcı	Değer	Açıklama
INDUSTRY_SILVER	12	Gümüş
INDUSTRY_SPECIALTY_CHEMICALS	13	Özel kimyasallar
INDUSTRY_STEEL	14	Çelik
İletişim servisleri		
INDUSTRY_ADVERTISING	51	Reklam ajansları
INDUSTRY_BROADCASTING	52	Radyo, televizyon vb. yayıncılığı
INDUSTRY_GAMING_MULTIMEDIA	53	Elektronik oyun ve multimedya
INDUSTRY_ENTERTAINMENT	54	Eğlence
INDUSTRY_INTERNET_CONTENT	55	İnternet içeriği ve bilgileri
INDUSTRY_PUBLISHING	56	Kitap, dergi vb. yayıncılığı
INDUSTRY_TELECOM	57	Telekomünikasyon hizmetleri
Tüketici - Konjonktürel		
INDUSTRY_APPAREL_MANUFACTURING	101	Kıyafet imalatı
INDUSTRY_APPAREL_RETAIL	102	Kıyafet perakendeciliği
INDUSTRY_AUTO_MANUFACTURERS	103	Otomobil üreticileri
INDUSTRY_AUTO_PARTS	104	Otomobil parçaları
INDUSTRY_AUTO_DEALERSHIP	105	Otomobil ve kamyon bayileri
INDUSTRY_DEPARTMENT_STORES	106	Çok katlı mağazalar
INDUSTRY_FOOTWEAR_ACCESSORIES	107	Ayakkabılar ve aksesuarlar
INDUSTRY_FURNISHINGS	108	Mobilya, demirbaşlar ve elektrikli aletler
INDUSTRY_GAMBLING	109	Kumar
INDUSTRY_HOME_IMPROV_RETAIL	110	Ev dekorasyonu perakendeciliği
INDUSTRY_INTERNET_RETAIL	111	İnternet perakendeciliği
INDUSTRY_LEISURE	112	Serbest zaman endüstrisi
INDUSTRY_LODGING	113	Konaklama
INDUSTRY_LUXURY_GOODS	114	Lüks mallar
INDUSTRY_PACKAGING_CONTAINERS	115	Ambalaj ve kaplar
INDUSTRY_PERSONAL_SERVICES	116	Bireysel hizmetler
INDUSTRY_RECREATIONAL_VEHICLES	117	Karavanlar
INDUSTRY_RESIDENT_CONSTRUCTION	118	Konut inşaatı

Tanımlayıcı	Değer	Açıklama
INDUSTRY_RESORTS_CASINOS	119	Tatil köyleri ve kumarhaneler
INDUSTRY_RESTAURANTS	120	Restoranlar
INDUSTRY_SPECIALTY_RETAIL	121	Özel perakendecilik
INDUSTRY_TEXTILE_MANUFACTURING	122	Tekstil imalatı
INDUSTRY_TRAVEL_SERVICES	123	Seyahat hizmetleri
Tüketici - İstikrarlı		
INDUSTRY_BEVERAGES_BREWERS	151	İçecekler - Bira imalathaneleri
INDUSTRY_BEVERAGES_NON_ALCO	152	İçecekler - Alkolsüz
INDUSTRY_BEVERAGES_WINERIES	153	İçecekler - Şarap imalathaneleri
INDUSTRY_CONFECTIONERS	154	Şekerlemeciler
INDUSTRY_DISCOUNT_STORES	155	İndirim mağazaları
INDUSTRY_EDUCATION_TRAINIG	156	Eğitim ve öğretim hizmetleri
INDUSTRY_FARM_PRODUCTS	157	Tarım ürünleri
INDUSTRY_FOOD_DISTRIBUTION	158	Gıda dağıtımı
INDUSTRY_GROCERY_STORES	159	Marketler
INDUSTRY_HOUSEHOLD_PRODUCTS	160	Ev ve kişisel ürünler
INDUSTRY_PACKAGED_FOODS	161	Paketli gıdalar
INDUSTRY_TOBACCO	162	Tütün
Enerji		
INDUSTRY_OIL_GAS_DRILLING	201	Petrol ve gaz sondajı
INDUSTRY_OIL_GAS_EP	202	Petrol ve gaz çıkarma ve işleme
INDUSTRY_OIL_GAS_EQUIPMENT	203	Petrol ve gaz ekipmanı ve hizmetleri
INDUSTRY_OIL_GAS_INTEGRATED	204	Petrol ve gaz - entegre hizmetler
INDUSTRY_OIL_GAS_MIDSTREAM	205	Petrol ve gaz - orta akım hizmetler
INDUSTRY_OIL_GAS_REFINING	206	Petrol ve gaz rafinasyonu ve pazarlama
INDUSTRY_THERMAL_COAL	207	Termal kömür
INDUSTRY_URANIUM	208	Uranyum
Finans		
INDUSTRY_EXCHANGE_TRADED_FUND	251	Borsa yatırım fonu

Tanımlayıcı	Değer	Açıklama
INDUSTRY_ASSETS_MANAGEMENT	252	Varlık yönetimi
INDUSTRY_BANKS_DIVERSIFIED	253	Bankalar - Çeşitli
INDUSTRY_BANKS_REGIONAL	254	Bankalar - Bölgesel
INDUSTRY_CAPITAL_MARKETS	255	Sermaye piyasaları
INDUSTRY_CLOSE_END_FUND_DEBT	256	Kapalı uçlu yatırım fonu - Borç
INDUSTRY_CLOSE_END_FUND_EQUITY	257	Kapalı uçlu yatırım fonu - Özsermaye
INDUSTRY_CLOSE_END_FUND_FOREIGN	258	Kapalı uçlu yatırım fonu - Yurt dışı kaynaklı
INDUSTRY_CREDIT_SERVICES	259	Kredi hizmetleri
INDUSTRY_FINANCIAL_CONGLOMERATE	260	Finansal holdingler
INDUSTRY_FINANCIAL_DATA_EXCHANGE	261	Finansal veriler ve borsa
INDUSTRY_INSURANCE_BROKERS	262	Sigorta brokerları
INDUSTRY_INSURANCE_DIVERSIFIED	263	Sigorta - Çeşitli
INDUSTRY_INSURANCE_LIFE	264	Sigorta - Hayat
INDUSTRY_INSURANCE_PROPERTY	265	Sigorta - Mal ve kaza
INDUSTRY_INSURANCE_REINSURANCE	266	Sigorta - Reasürans
INDUSTRY_INSURANCE_SPECIALTY	267	Sigorta - Özel
INDUSTRY_MORTGAGE_FINANCE	268	İpotek finansmanı
INDUSTRY_SHELL_COMPANIES	269	Paravan şirketler
Sağlık hizmeti		
INDUSTRY_BIOTECHNOLOGY	301	Biyoteknoloji
INDUSTRY_DIAGNOSTICS_RESEARCH	302	Tanı ve araştırma
INDUSTRY_DRUGS_MANUFACTURERS	303	İlaç üreticileri - genel
INDUSTRY_DRUGS_MANUFACTURERS_SPEC	304	İlaç üreticileri - Özel ve eşdeğer
INDUSTRY_HEALTHCARE_PLANS	305	Sağlık planları
INDUSTRY_HEALTH_INFORMATION	306	Sağlık bilgi hizmetleri
INDUSTRY_MEDICAL_FACILITIES	307	Sağlık tesisleri
INDUSTRY_MEDICAL_DEVICES	308	Tıbbi cihazlar
INDUSTRY_MEDICAL_DISTRIBUTION	309	Tıbbi dağıtım
INDUSTRY_MEDICAL_INSTRUMENTS	310	Tıbbi aletler ve malzemeler

Tanımlayıcı	Değer	Açıklama
INDUSTRY_PHARM_RETAILERS	311	İlaç perakendecileri
Sanayi		
INDUSTRY_AEROSPACE_DEFENSE	351	Havacılık ve savunma
INDUSTRY_AIRLINES	352	Hava yolları
INDUSTRY_AIRPORTS_SERVICES	353	Havaalanları ve hava hizmetleri
INDUSTRY_BUILDING_PRODUCTS	354	Yapı ürünleri ve ekipmanları
INDUSTRY_BUSINESS_EQUIPMENT	355	İş ekipmanları ve malzemeleri
INDUSTRY_CONGLOMERATES	356	Holdingler
INDUSTRY_CONSULTING_SERVICES	357	Danışmanlık hizmetleri
INDUSTRY_ELECTRICAL_EQUIPMENT	358	Elektrik ekipmanları ve parçaları
INDUSTRY_ENGINEERING_CONSTRUCTION	359	Mühendislik ve inşaat
INDUSTRY_FARM_HEAVY_MACHINERY	360	Tarım ve ağır iş makineleri
INDUSTRY_INDUSTRIAL_DISTRIBUTION	361	Sanayi dağıtımı
INDUSTRY_INFRASTRUCTURE_OPERATIONS	362	Altyapı hizmetleri
INDUSTRY_FREIGHT_LOGISTICS	363	Nakliye ve lojistik - entegre hizmetler
INDUSTRY_MARINE_SHIPPING	364	Deniz taşımacılığı
INDUSTRY_METAL_FABRICATION	365	Metal üretimi
INDUSTRY_POLLUTION_CONTROL	366	Kirlilik denetimi ve temizlenmesi
INDUSTRY_RAILROADS	367	Demiryolları
INDUSTRY_RENTAL_LEASING	368	Kiralama ve finansal kiralama hizmetleri
INDUSTRY_SECURITY_PROTECTION	369	Güvenlik ve koruma hizmetleri
INDUSTRY_SPEALITY_BUSINESS_SERVICES	370	Özel iş hizmetleri
INDUSTRY_SPEALITY_MACHINERY	371	Özel sanayi makineleri
INDUSTRY_STUFFING_EMPLOYMENT	372	İstihdam hizmetleri
INDUSTRY_TOOLS_ACCESSORIES	373	Araçlar ve aksesuarlar
INDUSTRY_TRUCKING	374	Kamyon taşımacılığı
INDUSTRY_WASTE_MANAGEMENT	375	Atık Yönetimi
Gayrimenkul		
INDUSTRY_REAL_ESTATE_DEVELOPMENT	401	Gayrimenkul - Geliştirme

Tanımlayıcı	Değer	Açıklama
INDUSTRY_REAL_ESTATE_DIVERSIFIED	402	Gayrimenkul - Çeşitli
INDUSTRY_REAL_ESTATE_SERVICES	403	Gayrimenkul hizmetleri
INDUSTRY_REIT_DIVERSIFIED	404	GYO - Çeşitli
INDUSTRY_REIT_HEALTCARE	405	GYO - Sağlık tesisleri
INDUSTRY_REIT_HOTEL_MOTEL	406	GYO - Otel ve motel
INDUSTRY_REIT_INDUSTRIAL	407	GYO - Sanayi
INDUSTRY_REIT_MORTAGE	408	GYO - İpotek
INDUSTRY_REIT_OFFICE	409	GYO - Ofis
INDUSTRY_REIT_RESIDENTAL	410	GYO - Konut
INDUSTRY_REIT_RETAIL	411	GYO - Perakendecilik
INDUSTRY_REIT_SPECIALITY	412	GYO - Özel
Teknoloji		
INDUSTRY_COMMUNICATION_EQUIPMENT	451	İletişim ekipmanı
INDUSTRY_COMPUTER_HARDWARE	452	Bilgisayar donanımı
INDUSTRY_CONSUMER_ELECTRONICS	453	Tüketici elektroniği
INDUSTRY_ELECTRONIC_COMPONENTS	454	Elektronik bileşenler
INDUSTRY_ELECTRONIC_DISTRIBUTION	455	Elektronik ve bilgisayar dağıtımı
INDUSTRY_IT_SERVICES	456	Bilgi teknolojisi hizmetleri
INDUSTRY_SCIENTIFIC_INSTRUMENTS	457	Bilimsel ve teknik aletler
INDUSTRY_SEMICONDUCTOR_EQUIPMENT	458	Yarı iletken ekipman ve materyaller
INDUSTRY_SEMICONDUCTORS	459	Yarı iletkenler
INDUSTRY_SOFTWARE_APPLICATION	460	Yazılım - Uygulama
INDUSTRY_SOFTWARE_INFRASTRUCTURE	461	Yazılım - Alt yapı
INDUSTRY_SOLAR	462	Güneş enerjisi
Kamu hizmetleri		
INDUSTRY_UTILITIES_DIVERSIFIED	501	Kamu hizmetleri - Çeşitli
INDUSTRY_UTILITIES_POWERPRODUCERS	502	Kamu hizmetleri - Bağımsız enerji üreticileri
INDUSTRY_UTILITIES_RENEWABLE	503	Kamu hizmetleri - Yenilenebilir enerji hizmeti

Tanımlayıcı	Değer	Açıklama
INDUSTRY_UTILITIES_REGULATED_ELECTRIC	504	Kamu hizmetleri - Düzenli elektrik hizmeti
INDUSTRY_UTILITIES_REGULATED_GAS	505	Kamu hizmetleri - Düzenli gaz hizmeti
INDUSTRY_UTILITIES_REGULATED_WATER	506	Kamu hizmetleri - Düzenli su hizmeti
INDUSTRY_UTILITIES_FIRST	501	Kamu hizmetlerinin numaralandırmasının başlangıcı. INDUSTRY_UTILITIES_DIVERSIFIED'a karşılık gelir.
INDUSTRY_UTILITIES_LAST	506	Kamu hizmetlerinin numaralandırmasının sonu. INDUSTRY_UTILITIES_REGULATED_WATER'a karşılık gelir.

Hesap Özellikleri

Mevcut hesap hakkında bilgi almak için birkaç fonksiyon bulunmaktadır: AccountInfoInteger(), AccountInfoDouble() ve AccountInfoString(). Fonksiyon parametreleri, karşılık gelen ENUM_ACCOUNT_INFO sayımlarından gelen değerleri alabilirler.

AccountInfoInteger() fonksiyonu için

ENUM_ACCOUNT_INFO_INTEGER

Tanıtıcı	Açıklama	Tip
ACCOUNT_LOGIN	Hesap numarası	long
ACCOUNT_TRADE_MODE	Hesap alım-satım modu	ENUM_ACCOUNT_TRADE_MODE
ACCOUNT_LEVERAGE	Hesap kaldıracı	long
ACCOUNT_LIMIT_ORDERS	İzin verilen maksimum bekleyen emir sayısı	int
ACCOUNT_MARGIN_SO_MODE	İzin verilen en küçük teminat için ayar modu	ENUM_ACCOUNT_STOPOUT_MODE
ACCOUNT_TRADE_ALLOWED	Mevcut hesap için izin verilen alım-satım	bool
ACCOUNT_TRADE_EXPERT	Uzman Danışman için izin	bool

Tanıtıcı	Açıklama	Tip
	verilen alım-satım	
ACCOUNT_MARGIN_MODE	Teminat hesaplama modu	ENUM_ACCOUNT_MARGIN_MODE
ACCOUNT_CURRENCY_DIGITS	İşlem sonuçlarının doğru bir şekilde gösterilmesi için gerekli olan hesap para birimindeki ondalık basamak sayısı	int
ACCOUNT_FIFO_CLOSE	Pozisyonların yalnızca FIFO kuralıyla kapatılabileceğini belirten bir işaret. Özellik değeri true olarak ayarlanırsa, her bir sembol	bool

Tanıtıcı	Açıklama	Tip
	de pozisyonlar en eskisinden başlayarak, açıldıkları sırayla kapatılır. Pozisyonların farklı bir sırayla kapatılmasının denenmesi durumunda, yatırımcı bir hata alır. Hedgingsiz pozisyon muhasebe modu olan hesaplar için (ACCOUNT_MARGIN_MODE!=ACCOUNT_MARGIN_MODE_RETAIL_HEDGING	

Tanıtıcı	Açıklama	Tip
	), özellik değeri her zaman false olur.	

AccountInfoDouble() fonksiyonu için

ENUM_ACCOUNT_INFO_DOUBLE

Tanıtıcı	Açıklama	Tip
ACCOUNT_BALANCE	Mevduat dövizi cinsinden hesap bakiyesi	double
ACCOUNT_CREDIT	Mevduat dövizi cinsinden hesap kredisi	double
ACCOUNT_PROFIT	Bir hesabın mevduat dövizi cinsinden mevcut karı	double
ACCOUNT_EQUITY	Mevduat dövizi cinsinden hesaptaki net varlık değeri	double
ACCOUNT_MARGIN	Mevduat dövizi	double

Tanıtıcı	Açıklama	Tip
	cinsinden kullanılan teminat	
ACCOUNT_MARGIN_FREE	Mevdut döviz birimi cinsinden serbest teminat	double
ACCOUNT_MARGIN_LEVEL	Yüzde olarak, hesap teminat seviyesi	double
ACCOUNT_MARGIN_SO_CALL	Teminat çağrısı seviyesi. ACCOUNT_MARGIN_SO_MODE'a bağlı olarak, yüzdelik şekilde veya hesap para birimi cinsiyle ifade edilir	double
ACCOUNT_MARGIN_SO_SO	Teminat StopOut seviyesi.	double

Tanıtıcı	Açıklama	Tip
	ACCOUNT_MARGIN_SO_MODE'a bağlı olarak, yüzdelik şekilde veya hesap para birimi cinsiyle ifade edilir	
ACCOUNT_MARGIN_INITIAL	Başlangıç teminatı. Hesap üzerindeki tüm bekleyen emirleri karşılamak için ayrılan teminat miktarı	double
ACCOUNT_MARGIN_MAINTENANCE	Sürdürme teminatı. Hesap üzerindeki tüm açık pozisyonları karşılamak için ayrılan	double

Tanıtıcı	Açıklama	Tip
	minimum teminat miktarı	
ACCOUNT_ASSETS	Hesaptaki mevcut aktifler	double
ACCOUNT_LIABILITIES	Hesaptaki mevcut pasifler	double
ACCOUNT_COMMISSION_BLOCKED	Hesapta bloke edilmiş mevcut komisyon miktarı	double

AccountInfoString() fonksiyonu için

ENUM_ACCOUNT_INFO_STRING

Tanıtıcı	Açıklama	Tip
ACCOUNT_NAME	Müşteri ismi	string
ACCOUNT_SERVER	Alım-satım sunucusunun ismi	string
ACCOUNT_CURRENCY	Mevduat dövizinin cinsi	string
ACCOUNT_COMPANY	Hesaba hizmet veren firmanın ismi	string

Bir alım-satım sunucusunda açılabilecek birkaç hesap tipi bulunmaktadır. Bir MQL5 programının çalıştırıldığı hesap tipi ENUM_ACCOUNT_TRADE_MODE sayımı ile bulunabilir.

ENUM_ACCOUNT_TRADE_MODE

Tanıtıcı	Açıklama
ACCOUNT_TRADE_MODE_DEMO	Deneme hesabı
ACCOUNT_TRADE_MODE_CONTEST	Yarışma hesabı
ACCOUNT_TRADE_MODE_REAL	Gerçek hesap

Net varlık, açık pozisyonları korumak için yeterli değilse, StopOut durumu, yani zorla durdurma durumu gerçekleşir. StopOut seviyesinin gerçekleşeceği minimum teminat seviyesi, yüzdelik olarak veya parasal terimlerle ayarlanmış olabilir. Hesap için modu öğrenmek amacıyla, ENUM_ACCOUNT_STOPOUT_MODE sayımını kullanın.

ENUM_ACCOUNT_STOPOUT_MODE

Tanıtıcı	Açıklama
ACCOUNT_STOPOUT_MODE_PERCENT	Yüzdelik olarak hesap StopOut modu
ACCOUNT_STOPOUT_MODE_MONEY	Parasal olarak hesap StopOut modu

ENUM_ACCOUNT_MARGIN_MODE

Tanıtıcı	Açıklama
ACCOUNT_MARGIN_MODE_RETAIL_NETTING	Yirmidört saat işlem gören (OTC) piyasalarda, netleştirme (netting modu: belli bir sembol üzerinde sadece bir açık pozison bulunabilir) ile açılan pozisyonları ifade etmek kullanılır. Teminat değeri sembol tipine göre hesaplanır (SYMBOL_TRADE_CALC_MODE).
ACCOUNT_MARGIN_MODE_EXCHANGE	Hisse senedi piyasaları için kullanılır. Teminat değeri sembol ayarlarında belirtilen iskontoya göre heaplanır. İskontolar borsada belirlenen değerden az olmamak koşuluyla aracı kurumlar tarafından ayarlanır.
ACCOUNT_MARGIN_MODE_RETAIL_HEDGING	Çoklu pozisyon imkanı tanınan piyasalarda kullanılır (hedge'li sistem, bir sembol üzerinde birden çok açık pozisyon bulunabilir). Teminat değeri sembol tipine göre hesaplanır (SYMBOL_TRADE_CALC_MODE) ve hedge'li teminat değeri hesaplamaya dahil edilir (SYMBOL_MARGIN_HEDGED).

Hesap bilgisinin özetini çıktılayan bir betik örneği.

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Firma ismi
   string company=AccountInfoString(ACCOUNT_COMPANY);
//--- Müşteri ismi
   string name=AccountInfoString(ACCOUNT_NAME);
//--- Hesap ismi
   long login=AccountInfoInteger(ACCOUNT_LOGIN);
//--- Sunucu ismi
   string server=AccountInfoString(ACCOUNT_SERVER);
//--- Mevduat dövizinin cinsi
   string currency=AccountInfoString(ACCOUNT_CURRENCY);
//--- Demo, yarışma veya gerçek hesap
   ENUM_ACCOUNT_TRADE_MODE account_type=(ENUM_ACCOUNT_TRADE_MODE)AccountInfoInteger(AC
//--- Şimdi sayım değerini anlaşılır bir şekle dönüştür
   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;
     }
//--- StopOut, yüzdelik olarak mı, yoksa parasal olarak mı ayarlanmış
   ENUM_ACCOUNT_STOPOUT_MODE stop_out_mode=(ENUM_ACCOUNT_STOPOUT_MODE)AccountInfoInteg
//--- StopOut ve teminat çağrısının gerçekleştiği seviye değerlerini al
   double margin_call=AccountInfoDouble(ACCOUNT_MARGIN_SO_CALL);
   double stop_out=AccountInfoDouble(ACCOUNT_MARGIN_SO_SO);
//--- Özet hesap bilgisini göster
   PrintFormat("'%s' #%d %s müşterisinin hesabı, '%s' firmasıyla '%s' sunucusunda açıl
               name,login,trade_mode,company,server);
   PrintFormat("Hesap para birimi - %s, teminat çağrısı ve StopOut seviyeleri %s cinsi
               currency,(stop_out_mode==ACCOUNT_STOPOUT_MODE_PERCENT)?"yüzdelik":" par
   PrintFormat("teminat çağrısı=%G, StopOut=%G",margin_call,stop_out);
  }
```

Sınama İstatistikleri

Sınama sona erdiğinde, alım-satım sonuçlarının istatistiklerine dair farklı parametreler hesaplanır. Parametrelerin değerleri, ENUM_STATISTICS sayımında parametre tanımlayıcısını belirledikten sonra, TesterStatistics() fonksiyonu kullanılarak elde edilebilir.

İstatistiklerin hesaplanması için iki farklı tipte (int ve double) parametre kullanılır, fonksiyon tüm değerleri double olarak döndürür. Aksi belirtilmedikçe double tipli istatistik değerleri, mevduat döviz birimi üzerinden ifade edilir.

ENUM_STATISTICS

Tanıtıcı	İstatistiksel parametrelerin tanımı	Tip
STAT_INITIAL_DEPOSIT	Başlangıç mevduat değeri	double
STAT_WITHDRAWAL	Hesaptan çekilen para miktarı	double
STAT_PROFIT	Sınamadan sonraki net kar, STAT_GROSS_PROFIT ve STAT_GROSS_LOSS değerlerinin toplamı (STAT_GROSS_LOSS her zaman sıfırdan küçük veya sıfıra eşittir)	double
STAT_GROSS_PROFIT	Toplam kar, tüm kazançlı (pozitif) alım-satım işlemlerinin toplamı. Değer, sıfırdan büyüktür veya sıfıra eşittir	double
STAT_GROSS_LOSS	Toplam kayıp, tüm negatif alım-satım işlemlerinin toplamı. Değer, sıfırdan küçüktür veya sıfıra eşittir	double
STAT_MAX_PROFITTRADE	Maksimum kar - tüm kazançlı işlemler içindeki en büyük değer. Değer, sıfırdan büyüktür veya sıfıra eşittir	double
STAT_MAX_LOSSTRADE	Maksimum kayıp - tüm kaybeden işlemlerin	double

Tanıtıcı	İstatistiksel parametrelerin tanımı	Tip
	arasındaki en küçük değer. Değer, sıfırdan küçüktür veya sıfıra eşittir	
STAT_CONPROFITMAX	Bir kazançlı işlem serisi içindeki maksimum kar. Değer, sıfırdan büyüktür veya sıfıra eşittir	double
STAT_CONPROFITMAX_TRADES	STAT_CONPROFITMAX (bir kazançlı işlem serisi içindeki maksimum kar) istatistiğini şekillendiren işlemlerin sayısı	int
STAT_MAX_CONWINS	En uzun kazançlı işlemler serisinin toplam karı	double
STAT_MAX_CONPROFIT_TRADES	En uzun kazançlı işlemler serisindeki işlem sayısı STAT_MAX_CONWINS	int
STAT_CONLOSSMAX	Bir kayıplı işlem serisi içindeki maksimum zarar. Değer, sıfırdan küçüktür veya sıfıra eşittir	double
STAT_CONLOSSMAX_TRADES	STAT_CONLOSSMAX (bir kaybeden işlem serisi içindeki maksimum zarar) istatistiğini şekillendiren işlemlerin sayısı	int
STAT_MAX_CONLOSSES	En uzun kaybeden işlem serisindeki toplam zarar	double
STAT_MAX_CONLOSS_TRADES	En uzun kaybeden işlem serisindeki işlemlerin sayısı STAT_MAX_CONLOSSES	int

Tanıtıcı	İstatistiksel parametrelerin tanımı	Tip
STAT_BALANCEMIN	Minimum bakiye değeri	double
STAT_BALANCE_DD	Parasal terimlerle bakiyedeki maksimum azalma. Alım-satım süreci içinde bakiyede birçok azalma gerçekleşebilir; burada en büyük değer alınır	double
STAT_BALANCEDD_PERCENT	Yüzdelik değerle bakiyedeki azalma, bakiyedeki maksimum azalmanın (STAT_BALANCE_DD) gerçekleşmesiyle kaydedilir.	double
STAT_BALANCE_DDREL_PERCENT	Yüzdelik olarak bakiyedeki maksimum azalma. Alım-satım süreci içinde bakiyede birçok azalma gerçekleşebilir, burada her biri için yüzdelik bazda göreli azalma değeri hesaplanır. En büyük değere dönüş yapılır	double
STAT_BALANCE_DD_RELATIVE	Parasal terimlerle bakiyedeki azalma, bakiyedeki maksimum yüzdelik azalmanın (STAT_BALANCE_DDREL_PERCENT) gerçekleşmesiyle kaydedilir.	double
STAT_EQUITYMIN	Minimum varlık değeri	double
STAT_EQUITY_DD	Parasal terimlerle varlıktaki maksimum azalma. Alım-satım süreci içinde varlıkta birçok azalma gerçekleşebilir; burada en büyük değer alınmaktadır	double

Tanıtıcı	İstatistiksel parametrelerin tanımı	Tip
STAT_EQUITYDD_PERCENT	Yüzdelik olarak azalma miktarı, varlıktaki maksimum parasal azalmanın (STAT_EQUITY_DD) gerçekleşmesiyle kaydedilir.	double
STAT_EQUITY_DDREL_PERCENT	Varlıktaki maksimum yüzdelik azalma. Alım-satım süreci içinde varlıkta birçok azalma gerçekleşebilir, burada her biri için yüzdelik bazda, göreli azalma değeri hesaplanır. En büyük değere dönüş yapılır	double
STAT_EQUITY_DD_RELATIVE	Parasal terimlerle varlıktaki azalma, varlıktaki maksimum yüzdelik azalma (STAT_EQUITY_DDREL_PERCENT) gerçekleştiğinde kaydedilir.	double
STAT_EXPECTED_PAYOFF	Beklenen kazanç	double
STAT_PROFIT_FACTOR	Kazanç faktörü, STAT_GROSS_PROFIT/STAT_GROSS_LOSS oranına eşittir. Eğer STAT_GROSS_LOSS=0 ise, kazanç faktörü DBL_MAX değerine eşittir	double
STAT_RECOVERY_FACTOR	Geri kazanım faktörü, STAT_PROFIT/STAT_BALANCE_DD oranına eşittir	double
STAT_SHARPE_RATIO	Sharpe oranı	double
STAT_MIN_MARGINLEVEL	Minimum teminat seviyesi değeri	double
STAT_CUSTOM_ONTESTER	OnTester() fonksiyonunun dönüş	double

Tanıtıcı	İstatistiksel parametrelerin tanımı	Tip
	yaptığı, hesaplanmış özel optimizasyon kriterinin değeri	
STAT_DEALS	İşlemlerin sayısı	int
STAT_TRADES	Yapılan işlemlerinin sayısı	int
STAT_PROFIT_TRADES	Kazançlı işlemler	int
STAT_LOSS_TRADES	Kaybeden işlemler	int
STAT_SHORT_TRADES	Kısa (satış) işlemler	int
STAT_LONG_TRADES	Uzun (alış) işlemler	int
STAT_PROFIT_SHORTTRADES	Kazançlı kısa işlemler	int
STAT_PROFIT_LONGTRADES	Kazançlı uzun işlemler	int
STAT_PROFITTRADES_AVGCON	Kazançlı bir işlem serisinin ortalama uzunluğu	int
STAT_LOSSTRADES_AVGCON	Kaybeden bir işlem serisinin ortalama uzunluğu	int
STAT_COMPLEX_CRITERION	Karmaşık optimizasyon kriteri	

Alım-Satım Sabitleri

Alım-satım stratejilerinin programlanması için kullanılan sabitler şu gruplara bölünmüştür:

- Tarihsel veritabanı özellikleri - bir sembole dair genel bilginin alınması;
- Emir özellikleri - alım-satım işlemlerine dair bilgilerin edinilmesi;
- Pozisyon özellikleri - mevcut pozisyonlarla ilgili bilginin edinilmesi;
- İşlem özellikleri - işlemlerle ilgili bilgilerin edinilmesi;
- Alım-satım işlem tipleri - mevcut alım-satım işlemlerinin açıklaması;
- Alım-satım faaliyet tipleri - muhtemel alım-satım faaliyeti tiplerinin açıklaması;
- Piyasa Derinliğinde alım-satım emirleri - istenen işlemin yönüne bağlı olarak, emirlerin ayrılması.

Tarihsel Veritabanı Özellikleri

Zaman serilerine erişim gerçekleştirilirken SeriesInfoInteger() fonksiyonu, ek sembol bilgisi alabilmek amacıyla kullanılır. İstenen bir özelliğin tanımlayıcısı, fonksiyona parametre olarak geçirilir. Tanımlayıcı, ENUM_SERIES_INFO_INTEGER sayımının değerlerinden biri olabilir.

ENUM_SERIES_INFO_INTEGER

Tanımlayıcı	Açıklama	Tip
SERIES_BARS_COUNT	Mevcut anda, belirli bir sembol-periyot için çubuk sayısı	long
SERIES_FIRSTDATE	Mevcut anda, belirli bir semboldeki-periyottaki ilk tarih	datetime
SERIES_LASTBAR_DATE	Semboldeki-periyottaki son çubuğun açılış zamanı	datetime
SERIES_SERVER_FIRSTDATE	Zaman aralığından bağımsız olarak, sembolün sunucuda bulunan geçmişindeki ilk tarih	datetime
SERIES_TERMINAL_FIRSTDATE	Zaman aralığından bağımsız olarak, sembolün terminalde bulunan geçmişindeki ilk tarih	datetime
SERIES_SYNCHRONIZED	Mevcut an için belirli bir semboldeki/periyottaki veri için senkronizasyon bayrağı	bool

Emir Özellikleri

Alım-satım işlemi istekleri emir olarak adlandırılır. Her emir bir dizi okuma amaçlı özelliğe sahiptir. Emirlerle ilgili bilgiler OrderGet...() ve HistoryOrderGet...() fonksiyonları ile elde edilebilir.

OrderGetInteger() ve HistoryOrderGetInteger() fonksiyonları için

ENUM_ORDER_PROPERTY_INTEGER

Tanıtıcı	Açıklama	Tip
ORDER_TICKET	Emir fişi. Her bir emir için atanan benzersiz tanıtıcı değer	long
ORDER_TIME_SETUP	Emir başlangıç zamanı	datetime
ORDER_TYPE	Emir tipi	ENUM_ORDER_TYPE
ORDER_STATE	Emir durumu	ENUM_ORDER_STATE
ORDER_TIME_EXPIRATION	Emir zaman-aşımı tipi	datetime
ORDER_TIME_DONE	Emrin gerçekleşme veya iptal edilme zamanı	datetime
ORDER_TIME_SETUP_MSC	01.01.1970 beri geçen milisaniyeler cinsinden emrin girilme zamanı	long
ORDER_TIME_DONE_MSC	01.01.1970 beri geçen milisaniyeler cinsinden emrin gerçekleşme/iptal edilme zamanı	long
ORDER_TYPE_FILLING	Emrin karşılanma tipi	ENUM_ORDER_TYPE_FILLING
ORDER_TYPE_TIME	Emrin yaşam süresi	ENUM_ORDER_TYPE_TIME

Tanıtıcı	Açıklama	Tip
ORDER_MAGIC	Emri veren Uzman Danışmanın tanımlayıcısı (her bir Uzman Danışmanın kendine has, benzersiz numarasını girmesi için tasarlanmıştır)	long
ORDER_REASON	Emrin veriliş sebebi veya kaynağı	ENUM_ORDER_REASON
ORDER_POSITION_ID	Pozisyon tanımlayıcısı. Emir erçekleştirildiği anda ayarlanır. Gerçekleştirilen her emir, mevcut bir pozisyonu açan veya değiştiren bir işlem ile sonuçlanır. Pozisyon tanımlayıcısı, gerçekleştirilen emir için gerçekleşme anında ayarlanır.	long
ORDER_POSITION_BY_ID	Emir ile kapama işlemi için kullanılacak ters pozisyonun tanımlayıcısı ORDER_TYPE_CLOSE_BY	long

OrderGetDouble() ve HistoryOrderGetDouble() fonksiyonları için

ENUM_ORDER_PROPERTY_DOUBLE

Tanıtıcı	Açıklama	Tip
ORDER_VOLUME_INITIAL	Emrin başlangıç hacmi	double
ORDER_VOLUME_CURRENT	Emrin mevcut hacmi	double
ORDER_PRICE_OPEN	Emirde belirtilen fiyat	double
ORDER_SL	Zarar Durdur değeri	double
ORDER_TP	Kar Al değeri	double
ORDER_PRICE_CURRENT	Emir verilen sembolün mevcut fiyatı	double
ORDER_PRICE_STOPLIMIT	StopLimit emri için Limitli emir fiyatı	double

OrderGetString() ve HistoryOrderGetString() fonksiyonları için

ENUM_ORDER_PROPERTY_STRING

Tanıtıcı	Açıklama	Tip
ORDER_SYMBOL	Emrin verildiği sembol	string
ORDER_COMMENT	Emir yorumu	string
ORDER_EXTERNAL_ID	Borsa türü piyasalar için emir tanımlayıcısı	string

OrderSend() fonksiyonu kullanarak bir alım-satım isteği gönderirken, bazı işlemler, emir tipinin belirtilmesini gerektirir. Emir tipi, MqlTradeRequest özel yapısının *type* alanında belirtilir.

ENUM_ORDER_TYPE

Tanıtıcı	Açıklama
ORDER_TYPE_BUY	Piyasa Alış emri
ORDER_TYPE_SELL	Piyasa Satış emri

Tanıtıcı	Açıklama
ORDER_TYPE_BUY_LIMIT	Limit Alış bekleyen emri
ORDER_TYPE_SELL_LIMIT	Limit Satış bekleyen emri
ORDER_TYPE_BUY_STOP	Stop Alış bekleyen emri
ORDER_TYPE_SELL_STOP	Stop Satış bekleyen emri
ORDER_TYPE_BUY_STOP_LIMIT	Emir fiyatına ulaşıldığında, StopLimit fiyatından bir Limit Alış emri yerleştirilir
ORDER_TYPE_SELL_STOP_LIMIT	Emir fiyatına ulaşıldığında, StopLimit fiyatından bir Limit Satış emri yerleştirilir
ORDER_TYPE_CLOSE_BY	Pozisyonu ters pozisyonla kapatmak için kullanılan emir

Her emir, durumunu tanımlayan bir statüye sahiptir. Bu bilgiyi almak için, OrderGetInteger() veya HistoryOrderGetInteger() fonksiyonlarını, ORDER_STATE şekillendiricisi ile kullanın. İzin verilen değerler ENUM_ORDER_STATE sayımında listelenmiştir.

ENUM_ORDER_STATE

Tanıtıcı	Açıklama
ORDER_STATE_STARTED	Emir kontrol edilmiş ama henüz aracı kurum tarafından kabul edilmemiş
ORDER_STATE_PLACED	Emir kabul edildi
ORDER_STATE_CANCELED	Emir müşteri tarafından iptal edildi
ORDER_STATE_PARTIAL	Emir kısmen işlendi
ORDER_STATE_FILLED	Emir bütünüyle işlendi
ORDER_STATE_REJECTED	Emir reddedildi
ORDER_STATE_EXPIRED	Emir, zaman-aşımına uğradı
ORDER_STATE_REQUEST_ADD	Emir işleniyor (alım-satım sistemine kaydediliyor)
ORDER_STATE_REQUEST_MODIFY	Emir şekillendiriliyor (parametreleri değiştiriliyor)
ORDER_STATE_REQUEST_CANCEL	Emir siliniyor (alım-satım sisteminden siliniyor)

Geçerli zamanda (yürürlükteki zaman) işlem gerçekleştirmek için bir işlem talebi gönderirken, fiyat ve gerekli alım/satım hacmi belirtilmelidir. Ayrıca, finansal piyasaların, talep edilen hacmin tamamının belirli bir finansal enstrüman için istenen fiyatta mevcut olduğuna dair hiçbir garanti vermediğini unutmayın. Bu nedenle, gerçek zamanlı ticaret işlemleri, fiyatla ve hacimle ilişkili işlem gerçekleştirme modları kullanılarak düzenlenir. Modlar veya işlem gerçekleştirme politikaları, fiyatın değiştiği veya istenen hacmin **o anda** tam olarak karşılanamadığı durumlar için kuralları tanımlar.

Fiyatla ilişkili işlem gerçekleştirme modu, ENUM_SYMBOL_TRADE_EXECUTION sayımındaki bayrak kombinasyonunu içeren SYMBOL_TRADE_EXEMODE sembol özelliğinden elde edilebilir.

İşlem gerçekleştirme modu	Açıklama	ENUM_SYMBOL_TRADE_EXECUTION'daki değer
İşlem gerçekleştirme modu (İşlem Gerçekleştirme Talebi)	Brokerdan alınan fiyattan bir piyasa emri gerçekleştirme. Belirli bir piyasa emrinin fiyatları, emir gönderilmeden önce brokerdan talep edilir. Fiyatlar alındıktan sonra, elde edilen fiyattan emrin gerçekleştirilmesi onaylanabilir veya reddedilebilir.	SYMBOL_TRADE_EXECUTION_REQUEST
Anında İşlem Gerçekleştirme (Anında İşlem Gerçekleştirme)	Belirtilen fiyatta	SYMBOL_TRADE_EXECUTION_INSTANT

İşlem gerçekleştirme modu	Açıklama	ENUM_SYMBOL_TRADE_EXECUTION'daki değer
	n anında bir piyasa emri gerçekleştirme. Gerçekleştirilmek üzere bir işlem talebi gönderirken, platform otomatik olarak mevcut fiyatları emre ekler. • Broker fiyatı kabul eder	

İşlem gerçekleştirme modu	Açıklama	ENUM_SYMBOL_TRADE_EXECUTION'daki değer
	se, emir gerçekleştirilir. • Broker fiyatı kabul etmezse	

İşlem gerçekleştirme modu	Açıklama	ENUM_SYMBOL_TRADE_EXECUTION'daki değer
	, bir "Requote" gönderilir - broker bu emri inceler ve kabul etti	

İşlem gerçekleştirme modu	Açıklama	ENUM_SYMBOL_TRADE_EXECUTION'daki değer
	rilebilecceğifiyatlarıgeridöndürür.	
Piyasa Fiyatından İşlem Gerçekleştirme (Piyasa Fiyatından İşlem Gerçekleştirme)	Brokeryatırımcıylaherhangi bir ek iletişim yapmadan emir gerçekleştirme fiyatı	SYMBOL_TRADE_EXECUTION_MARKET

İşlem gerçekleştirme modu	Açıklama	ENUM_SYMBOL_TRADE_EXECUTION'daki değer
	hakkında karar verir. Emrin bu modda gönderilmesi, o anki fiyattan emrin gerçekleşmesine öncedeN izin verilmesi anlamına gelir.	
Exchange İşlem Gerçekleştirme (Exchange İşlem Gerçekleştirme)	Ticaret işlemleri mevcut piyasa fiyatları üzerinden gerçekleştirilir.	SYMBOL_TRADE_EXECUTION_EXCHANGE

Hacim yerine getirme politikası ORDER_TYPE_FILLING emir özelliğinde belirtilir ve yalnızca ENUM_ORDER_TYPE_FILLING sayımındaki değerleri içerebilir

Hacim yerine getirme politikası	Açıklama	ENUM_ORDER_TYPE_FILLING'deki değer
Kalanı İptal Et (Fill or Kill, FOK)	Bir emir yalnızca belirtil	ORDER_FILLING_FOK

Hacim yerine getirme politikası	Açıklama	ENUM_ORDER_TYPE_FILLING'deki değer
	en hacimde gerçekleştirilebilir. Finansal enstrüman için piyasada gerekli miktarda hacim mevcut değilse, emir gerçekleşmez. İstenen hacim mevcut birkaç fiyatı içerebilir. FOK emirlerini kullanma imkanı ticaret sunucusu tarafından belirlenir.	

Hacim yerine getirme politikası	Açıklama	ENUM_ORDER_TYPE_FILLING'deki değer
Anında ya da İptal Et (Immediate or Cancel, IOC)	Yatırımcı, emirde belirtilen dahilinde piyasada bulunan maksimum hacimde işlem gerçekleştirmeyi kabul eder. Talebin tam olarak karşılanaması durumunda, emir var olan hacimde gerçekleştirilecek ve kalan hacim iptal edilecektir. IOC emirlerini kullanma	ORDER_FILLING_IOC

Hacim yerine getirme politikası	Açıklama	ENUM_ORDER_TYPE_FILLING'deki değer
	imkanı ticaret sunucusu tarafından belirlenir.	
Pasife yerleştir (Book Or Cancel, BOC)	BOC politikası, emrin yalnızca Piyasa Derinliğine yerleştirilebileceğini ve hemen gerçekleştirilemeyeceğini belirtir. Emir yerleştirildiği anda hemen yerine getirilebiliyorsa bu emir iptal edilir. Aslında bu politika, yerleştirilen emrin	ORDER_FILLING_BOC

Hacim yerine getirme politikası	Açıklama	ENUM_ORDER_TYPE_FILLING'deki değer
	fiyatının mevcut piyasadan daha kötü olacağını garanti eder. BOC emirleri pasif ticaret yapmak için kullanılır: emrin yerleştirildiği anda yerine getirilmeyeceği ve böylece mevcut likiditeyi etkilemeyeceği garanti edilir. Yalnızca limit ve stop limit emirleri destekl	

Hacim yerine getirme politikası	Açıklama	ENUM_ORDER_TYPE_FILLING'deki değer
	enmektedir (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT, ORDER_TYPE_SELL_STOP_LIMIT).	
Dönüş	Hacmin kısmi olarak yerine getirilmesi durumunda, hacmi artmış emir iptal edilmez, daha sonrası için işlenir. Piyasa Fiyatından İşlem Gerçekleştirm	ORDER_FILLING_RETURN

Hacim yerine getirme politikası	Açıklama	ENUM_ORDER_TYPE_FILLING'deki değer
	e modunda dönüş emirlerine izin verilmez (piyasa fiyatından işlem gerçekleştirme — SYMBOL_TRADE_EXECUTION_MARKET).	

OrderSend() fonksiyonunu kullanarak bir ticaret talebi gönderirken, *type_filling* alanında, yani özel MqlTradeRequest yapısında gerekli hacim yerine getirme politikası ayarlanabilir. ENUM_ORDER_TYPE_FILLING sayımındaki değerler kullanılabilir. Belirli bir etkin/tamamlanmış emirdeki özellik değerini elde etmek için, ORDER_TYPE_FILLING düzenleyicisiyle OrderGetInteger() veya HistoryOrderGetInteger() fonksiyonunu kullanın.

Geçerli işlem gerçekleştirme zamanıyla bir emir göndermeden önce, ORDER_TYPE_FILLING değerinin (hacimle ilişkili işlem gerçekleştirme tipi) doğru ayarlanması için, SYMBOL_FILLING_MODE özellik değerini elde etmek adına her bir finansal enstrümanla SymbolInfoInteger() fonksiyonunu (bir bayrak kombinasyonu şeklinde sembol için izin verilen hacimle ilişkili işlem gerçekleştirme tiplerini gösterir) kullanabilirsiniz. ORDER_FILLING_RETURN tipi, "Piyasa fiyatından işlem gerçekleştirme" modu (SYMBOL_TRADE_EXECUTION_MARKET) dışında her zaman etkindir.

İşlem gerçekleştirme moduna bağlı olarak hacim yerine getirme tiplerinin kullanımı aşağıdaki tablo halinde gösterilebilir:

İşlem Gerçekleştirme Tipi\Hacim Yerine Getirme Politikası	Kalanı İptal Et (FOK ORDER_FILLING_FOK)	Anında ya da İptal Et (IOC ORDER_FILLING_IOC)	Dönüş (Return ORDER_FILLING_RETURN)
Anında İşlem Gerçekleştirme (SYMBOL_TRADE_EXECUTION_INSTANT)	+ (sembol ayarından bağımsız olarak)	+ (sembol ayarından bağımsız olarak)	+ (her zaman)

İşlem Gerçekleştirme Tipi\Hacim Yerine Getirme Politikası	Kalanı İptal Et (FOK ORDER_FILLING_FOK)	Anında ya da İptal Et (IOC ORDER_FILLING_IOC)	Dönüş (Return ORDER_FILLING_RETURN)
İşlem Gerçekleştirme Talebi SYMBOL_TRADE_EXECUTION_REQUEST	+ (sembol ayarından bağımsız olarak)	+ (sembol ayarından bağımsız olarak)	+ (her zaman)
Piyasa Fiyatından İşlem Gerçekleştirme SYMBOL_TRADE_EXECUTION_MARKET	+ (sembol ayarlarından ayarlanır)	+ (sembol ayarlarından ayarlanır)	- (sembol ayarlarından bağımsız olarak devre dışı)
Exchange İşlem Gerçekleştirme SYMBOL_TRADE_EXECUTION_EXCHANGE	+ (sembol ayarlarından ayarlanır)	+ (sembol ayarlarından ayarlanır)	+ (her zaman)

Bekleyen emirler olması durumunda, bu tür emirler gönderme sırasında gerçekleştirilmeleri üzere tasarlanmadığından, işlem gerçekleştirme tipinden (SYMBOL_TRADE_EXEMODE) bağımsız olarak ORDER_FILLING_RETURN hacim yerine getirme tipi kullanılmalıdır. Bekleyen emirleri kullanırken, yatırımcı, emirdeki koşulların işlem için karşılandığında, brokerın borsa tarafından desteklenen hacim yerine getirme tipini kullanacağını peşinen kabul eder.

OrderSend() fonksiyonu ile emir gönderirken, emir geçerlilik periyodu, MqlTradeRequest özel yapısının *type_time* alanında ayarlanabilir. Bunun için ENUM_ORDER_TYPE_TIME sayımının değerleri kullanılır. Bu özelliğin değerini elde etmek amacıyla OrderGetInteger() veya HistoryOrderGetInteger() fonksiyonları, ORDER_TYPE_TIME şekillendiricisi ile kullanılır.

ENUM_ORDER_TYPE_TIME

Tanıtıcı	Açıklama
ORDER_TIME_GTC	İptale kadar geçerli emir
ORDER_TIME_DAY	Günlük emir
ORDER_TIME_SPECIFIED	Tarihli emir
ORDER_TIME_SPECIFIED_DAY	Emir, belirtilen günün 23:59:59 saatine kadar aktif olacaktır. Eğer belirtilen zaman alım-satım seansının dışında kalıyorsa, emir en yakın alım-satım seansının başında zaman aşımına uğrayacaktır.

Gerçekleştirilen emirle ilgili sebep verisi ORDER_REASON özelliğinde tutulur. Emirler, mobil uygulamalardan veya MQL5 programları, StopOut olayı, vb. gibi sebeplerle verilebilir. ORDER_REASON özelliğinin muhtemel değerleri ENUM_ORDER_REASON sayımında tanımlanmıştır.

ENUM_ORDER_REASON

Tanıtıcı	Açıklama
ORDER_REASON_CLIENT	Emir masaüstü terminali ile verilmiş
ORDER_REASON_MOBILE	Emir bir mobil uygulama ile verilmiş
ORDER_REASON_WEB	Emir bir mobil uygulama ile verilmiş
ORDER_REASON_EXPERT	Emir, bir MQL5 programı ile (Uzman Danışman veya betik) verilmiş
ORDER_REASON_SL	Emir, bir Zarar Durdur emrinin tetiklenmesi sonucunda oluşturulmuş
ORDER_REASON_TP	Emir, bir Kar Al emrinin tetiklenmesi sonucunda oluşturulmuş
ORDER_REASON_SO	Emir, bir StopOut olayının tetiklenmesi sonucunda oluşturulmuş

Pozisyon Özellikleri

Alım-satım işlemlerinin gerçekleştirilmesi; bir pozisyonun açılması, pozisyonun hacminin veya yönünün değiştirilmesi veya kapatılması ile sonuçlanır. Alım-satım işlemleri, alım-satım isteği şeklinde OrderSend() fonksiyonu ile gönderilen emirlere göre yönetilir. Her finansal menkul değer için sadece bir açık pozisyon mümkündür. Pozisyonlar PositionGet...() fonksiyonları ile okunabilen bir dizi özelliğe sahiptirler.

PositionGetInteger() fonksiyonu için

ENUM_POSITION_PROPERTY_INTEGER

Tanıtıcı	Açıklama	Tip
POSITION_TICKET	Pozisyon fişi. Açılan her yeni pozisyon için benzersiz bir numara atanır. Bu genellikle pozisyonu açmak için kullanılan emirle eşleşir (fiş sunucu tarafından servis işlemleri için değiştirilmediyse. Örneğin, pozisyon yenileme sırasında swap ücretlendirmesi yapıldığında). Pozisyon açmak için kullanılan bir emri bulmak için POSITION_IDENTIFIER özelliğini kullanın. POSITION_TICKET değeri MqlTradeRequest::position değerine karşılık gelir.	long
POSITION_TIME	Pozisyon açılış zamanı	datetime
POSITION_TIME_MSC	01.01.1970 tarihinden beri geçen	long

Tanıtıcı	Açıklama	Tip
	milisaniyeler şeklinde pozisyon açılış zamanı	
POSITION_TIME_UPDATE	01.01.1970 tarihinden beri geçen saniyeler şeklinde pozisyon değişim zamanı	datetime
POSITION_TIME_UPDATE_MSC	01.01.1970 tarihinden beri geçen milisaniyeler şeklinde pozisyon değişim zamanı	long
POSITION_TYPE	Pozisyon tipi	ENUM_POSITION_TYPE
POSITION_MAGIC	Pozisyon magic number (bakınız ORDER_MAGIC)	long
POSITION_IDENTIFIER	Pozisyon tanımlayıcısı her yeni pozisyon için atanan benzersiz bir numaradır. Pozisyonun işleyişini değiştirmez ve pozisyonu açmak için kullanılan emre karşılık gelir. Pozisyon tanımlayıcısı emri açmak, değiştirmek veya kapatmak için kullanılan her emir(ORDER_POSITION_ID) ve işlem (DEAL_POSITION_ID) için belirtilir. Pozisyonla ilgili işlem ve emirleri bulmak için bu özeliği kullanın.	long

Tanıtıcı	Açıklama	Tip
	Netleştirme modunda bir pozisyonu ters çevirirken (bir giriş veya çıkış işlemiyle) POSITION_IDENTIFIER değeri değişmez. Ama, POSITION_TICKET değeri ters çevirme işlemini yaptıran emrin fişi ile değiştirilir. Pozisyonlar hedge modunda ters çevrilemez.	
POSITION_REASON	Pozisyonun açılma sebebi	ENUM_POSITION_REASON

PositionGetDouble() fonksiyonu için

ENUM_POSITION_PROPERTY_DOUBLE

Tanıtıcı	Açıklama	Tip
POSITION_VOLUME	Pozisyon Hacmi	double
POSITION_PRICE_OPEN	Pozisyon açılış fiyatı	double
POSITION_SL	Açık Pozisyon için Zarar Durdur seviyesi	double
POSITION_TP	Açık pozisyon için Kar Al seviyesi	double
POSITION_PRICE_CURRENT	Pozisyon sembolünün mevcut fiyatı	double
POSITION_SWAP	Birikmiş swap	double
POSITION_PROFIT	Mevcut kar	double

PositionGetString() fonksiyonu için

ENUM_POSITION_PROPERTY_STRING

Tanıtıcı	Açıklama	Tip
POSITION_SYMBOL	Pozisyon sembolü	string
POSITION_COMMENT	Pozisyon yorumu	string

Açık bir pozisyonun yönü (alış veya satış), ENUM_POSITION_TYPE sayımının değerleri ile tanımlanır. Açık bir pozisyonun tipi hakkında bilgi almak için PositionGetInteger() fonksiyonunu POSITION_TYPE şekillendiricisi ile kullanın.

ENUM_POSITION_TYPE

Tanıtıcı	Açıklama
POSITION_TYPE_BUY	Alış işlemi
POSITION_TYPE_SELL	Satış

Açılan pozisyonla ilgili sebep verisi POSITION_REASON özelliğinde tutulur. Pozisyon, bir mobil uygulama ile yerleştirilen bir emrin sonucu olarak gerçekleştirilmiş POSITION_REASON özelliğinin muhtemel değerleri ENUM_POSITION_REASON sayımında tanımlanmıştır.

ENUM_POSITION_REASON

Tanıtıcı	Açıklama
POSITION_REASON_CLIENT	Pozisyon, masaüstü terminali ile yerleştirilen bir emrin sonucu olarak açılmış
POSITION_REASON_MOBILE	Pozisyon, bir mobil uygulama ile yerleştirilen bir emrin sonucu olarak açılmış
POSITION_REASON_WEB	Pozisyon, bir çevrim içi platformu ile yerleştirilen bir emrin sonucu olarak açılmış
POSITION_REASON_EXPERT	Pozisyon, bir MQL5 programı ile (Uzman Danışman veya betik) yerleştirilen bir emrin sonucu olarak açılmış

İşlem Özellikleri

İşlem, alım-satım isteği taşıyan bir emrin temelinde gerçekleşen bir alım satım eyleminin yansımasıdır. Her bir alım-satım işlemi, kendisi hakkında bilgi alınmasını sağlayan özellikler tarafından tanımlanır. Bu özelliklerin değerlerinin okunmasında, karşılık gelen sayım değerine dönüş yapan HistoryDealGet...() tipi fonksiyonlar kullanılır.

HistoryDealGetInteger() fonksiyonu için

ENUM_DEAL_PROPERTY_INTEGER

Tanıtıcı	Açıklama	Tip
DEAL_TICKET	İşlem fişi. Her bir işlem için atanan benzersiz tanıtıcı değer	long
DEAL_ORDER	İşlemin emir numarası	long
DEAL_TIME	İşlem zamanı	datetime
DEAL_TIME_MSC	01.01.1970 tarihinden beri geçen milisaniyeler şeklinde, işlemin gerçekleştirilme zamanı	long
DEAL_TYPE	İşlem tipi	ENUM_DEAL_TYPE
DEAL_ENTRY	Faaliyet yönü - piyasaya giriş, piyasadan çıkış veya dönüş	ENUM_DEAL_ENTRY
DEAL_MAGIC	Faaliyet için magic number (bakınız, ORDER_MAGIC)	long
DEAL_REASON	İşlemin gerçekleştirilme sebebi veya kaynağı	ENUM_DEAL_REASON
DEAL_POSITION_ID	Pozisyonun tanımlayıcısı, pozisyonun açılışı, değiştirilmesi ve kpanışı sırasında kullanılır. Her bir pozisyonun, aktif olduğu süre içinde, sembol için gerçekleştirilmiş tüm işlemlere atanan benzersiz bir tanımlayıcısı (kimliği) vardır.	long

HistoryDealGetDouble() fonksiyonu için

ENUM_DEAL_PROPERTY_DOUBLE

Tanıtıcı	Açıklama	Tip
DEAL_VOLUME	İşlemin hacmi	double
DEAL_PRICE	İşlem fiyatı	double
DEAL_COMMISSION	İşlem komisyonu	double
DEAL_SWAP	Kapanıştaki birikmiş swap	double
DEAL_PROFIT	İşlem karı	double

Tanıtıcı	Açıklama	Tip
DEAL_FEE	İşlem gerçekleştikten hemen sonra tahsil edilen işlem yapma ücreti	double
DEAL_SL	Zararı Durdur seviyesi • Giriş ve tersine çevirme işlemleri, pozisyonun açıldığı veya tersine çevrildiği esas emirdeki Zararı Durdur değerini kullanır • Çıkış işlemleri, pozisyonun kapatıldığı andaki Zararı Durdur değerini kullanır	double
DEAL_TP	Kârı Al seviyesi • Giriş ve tersine çevirme işlemleri, pozisyonun açıldığı veya tersine çevrildiği esas emirdeki Kârı Al değerini kullanır • Çıkış işlemleri, pozisyonun kapatıldığı andaki Kârı Al değerini kullanır	double

HistoryDealGetString() fonksiyonu için

ENUM_DEAL_PROPERTY_STRING

Tanıtıcı	Açıklama	Tip
DEAL_SYMBOL	İşlem sembolü	string
DEAL_COMMENT	İşlem yorumu	string
DEAL_EXTERNAL_ID	Borsa türü piyasalar için işlem tanımlayıcısı	string

Her işlem bir tip ile nitelendirilir, izin verilen değerler ENUM_DEAL_TYPE sayımında listelenmiştir. İşlem tipi ile ilgili bilgi edinmek için, HistoryDealGetInteger() fonksiyonunu DEAL_TYPE şekillendiricisi ile kullanın.

ENUM_DEAL_TYPE

Tanıtıcı	Açıklama
DEAL_TYPE_BUY	Alış işlemi
DEAL_TYPE_SELL	Satış
DEAL_TYPE_BALANCE	Bakiye

Tanıtıcı	Açıklama
DEAL_TYPE_CREDIT	Kredi
DEAL_TYPE_CHARGE	Ek ücret
DEAL_TYPE_CORRECTION	Düzeltme
DEAL_TYPE_BONUS	Bonus
DEAL_TYPE_COMMISSION	Ek komisyon
DEAL_TYPE_COMMISSION_DAILY	Günlük komisyon
DEAL_TYPE_COMMISSION_MONTHLY	Aylık komisyon
DEAL_TYPE_COMMISSION_AGENT_DAILY	Günlük ajans komisyonu
DEAL_TYPE_COMMISSION_AGENT_MONTHLY	Aylık ajans komisyonu
DEAL_TYPE_INTEREST	Faiz oranı
DEAL_TYPE_BUY_CANCELED	İptal edilmiş alım işlemi. Önceden gerçekleştirilmiş bir alım işleminin iptal edildiği bir durumla karşılaşılabilir. Bu durumda, daha önce gerçekleştirilmiş işlem (DEAL_TYPE_BUY), DEAL_TYPE_BUY_CANCELED tipine dönüştürülür ve ondan elde edilen kar/zarar sıfırlanır. Önceden elde edilen kar/zarar, ayrı bir bakiye işlemi ile ücretlendirilir/çekilir
DEAL_TYPE_SELL_CANCELED	İptal edilmiş satış işlemi. Önceden gerçekleştirilmiş bir satış işleminin iptal edildiği bir durumla karşılaşılabilir. Bu durumda, daha önce gerçekleştirilmiş işlem (DEAL_TYPE_SELL), DEAL_TYPE_SELL_CANCELED tipine dönüştürülür ve ondan elde edilen kar/zarar sıfırlanır. Önceden elde edilen kar/zarar, ayrı bir bakiye işlemi ile ücretlendirilir/çekilir
DEAL_DIVIDEND	Hisse işlemleri
DEAL_DIVIDEND_FRANKED	Damgalı (vergilendirilemeyen) hisse işlemleri
DEAL_TAX	Vergi ücretleri

İşlemler sadece ENUM_DEAL_TYPE içinde ayarlanmış tiplerine göre ayrılmazlar, ayrıca pozisyon değiştirme yöntemleri ile de ayrılırlar. Bu yöntem basit bir pozisyon açılışı veya daha önceden açılmış pozisyonların (piyasa girişi) birikimi olabilir; karşılık gelen hacimde ters işlem ile pozisyon kapama (piyasa çıkışı) veya - ters yönlü işlem daha önce açılan pozisyonun hacmini karşılıyorsa - pozisyonu ters çevirme şeklinde de olabilir.

Tüm bu durumlar ENUM_DEAL_ENTRY sayımının değerleri ile tarif edilir. Bir işlemle ilgili bu tip bir bilgiyi almak için, HistoryDealGetInteger() fonksiyonunu DEAL_ENTRY şekillendiricisi ile kullanın.

ENUM_DEAL_ENTRY

Tanıtıcı	Açıklama
DEAL_ENTRY_IN	Piyasaya giriş
DEAL_ENTRY_OUT	Piyasadan çıkış
DEAL_ENTRY_INOUT	Ters
DEAL_ENTRY_OUT_BY	Pozisyonu, ters pozisyon kullanarak kapa

Gerçekleştirilen işlemle ilgili sebep verisi DEAL_REASON özelliğinde tutulur. İşlemler, mobil uygulamalardan veya MQL5 programından verilen bir emrin tetiklenmesiyle oluşabileceği gibi, StopOut olayı, varyasyon teminatının hesaplanması, vb. gibi sebeplerle oluşabilir. DEAL_REASON özelliğinin muhtemel değerleri ENUM_DEAL_REASON sayımında tanımlanmıştır. Bakiye, kredi, komisyon, vb. sonucu oluşan ticari olmayan işlemlerde sebep olarak 'DEAL_REASON_CLIENT' belirtilir.

ENUM_DEAL_REASON

Tanıtıcı	Açıklama
DEAL_REASON_CLIENT	İşlem, masaüstü terminali ile yerleştirilen bir emrin sonucu olarak gerçekleştirilmiş
DEAL_REASON_MOBILE	İşlem, bir mobil uygulama ile yerleştirilen bir emrin sonucu olarak gerçekleştirilmiş
DEAL_REASON_WEB	İşlem, bir çevrim içi platformu ile yerleştirilen bir emrin sonucu olarak gerçekleştirilmiş
DEAL_REASON_EXPERT	İşlem, bir MQL5 programı ile (Uzman Danışman veya betik) yerleştirilen bir emrin sonucu olarak gerçekleştirilmiş
DEAL_REASON_SL	İşlem, bir Zarar Durdur emrinin tetiklenmesi sonucunda gerçekleştirilmiş
DEAL_REASON_TP	İşlem, bir Kar Al emrinin tetiklenmesi sonucunda gerçekleştirilmiş
DEAL_REASON_SO	İşlem, bir StopOut olayının tetiklenmesi sonucunda gerçekleştirilmiş
DEAL_REASON_ROLLOVER	İşlem bir devir sonucunda gerçekleştirilmiş
DEAL_REASON_VMARGIN	İşlem, varyasyon teminatının ücretlendirilmesinin ardından gerçekleştirilmiş
DEAL_REASON_SPLIT	İşlem, üstünde açık pozisyon bulunan bir işleme dair fiyat düşürme anonsu sonrasında gerçekleştirilmiş

Alım-Satım İşlem Tipleri

Alım-satım işlemleri pozisyon açmak için OrderSend() fonksiyonu kullanılarak emirlerin gönderilmesiyle gerçekleştirilir. Bu işlemler bekleyen emirlerin yerleştirilmesi, değiştirilmesi ve silinmesinde de kullanılır. Her alım-satım emri, istenen işlemin tipini gösterir. Alım-satım işlemleri ENUM_TRADE_REQUEST_ACTIONS sayımı içinde tarif edilmiştir.

ENUM_TRADE_REQUEST_ACTIONS

Tanıtıcı	Açıklama
TRADE_ACTION_DEAL	Anlık uygulama için belirlenen parametrelerle bir alım-satım emrini işleme koyar (piyasa emri)
TRADE_ACTION_PENDING	Bir alım-satım emrini, belirtilen koşullar altında uygulanacak şekilde işleme koyar (bekleyen emir)
TRADE_ACTION_SLTP	Bir açık pozisyon için Zarar Durdur ve Kar Al değerlerini şekillendirir
TRADE_ACTION_MODIFY	Daha önce verilmiş bir emrin parametrelerini şekillendirir
TRADE_ACTION_REMOVE	Önceden verilmiş bekleyen emri siler
TRADE_ACTION_CLOSE_BY	Pozisyonu, ters pozisyon kullanarak kapa

Alış pozisyonu açmak için TRADE_ACTION_DEAL işleminin bir örneği:

```
#define EXPERT_MAGIC 123456   // uzmanın tanıtıcı değeri (MagicNumber)
//+------------------------------------------------------------------+
//| Alış pozisyonu açılışı                                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- alım-satım isteğininin ve istek sonucunun bildirimini yap ve başlat
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- istek parametreleri
   request.action   =TRADE_ACTION_DEAL;                     // alım-satım işleminin tü
   request.symbol   =Symbol();                              // sembol
   request.volume   =0.1;                                   // 0.1 lotluk hacim lot
   request.type     =ORDER_TYPE_BUY;                        // emir türü
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_ASK); // açılış fiyatı
   request.deviation=5;                                     // izin verilen slipaj mil
   request.magic    =EXPERT_MAGIC;                          // emrin tanıtıcı değeri
//--- emri gönder
   if(!OrderSend(request,result))
      PrintFormat("OrderSend hatası %d",GetLastError());     // emir gönderilemiyorsa
//--- işlem bilgileri
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Satış pozisyonu açmak için TRADE_ACTION_DEAL işleminin bir örneği:

```
#define EXPERT_MAGIC 123456   // uzmanın tanıtıcı değeri (MagicNumber)
//+------------------------------------------------------------------+
//| Satış pozisyonu açılışı                                          |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- alım-satım isteğininin ve istek sonucunun bildirimini yap ve başlat
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- istek parametreleri
   request.action   =TRADE_ACTION_DEAL;                     // alım-satım işleminin tü
   request.symbol   =Symbol();                              // sembol
   request.volume   =0.2;                                   // 0.2 lotluk hacim
   request.type     =ORDER_TYPE_SELL;                       // emir türü
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_BID); // açılış fiyatı
   request.deviation=5;                                     // izin verilen slipaj mil
   request.magic    =EXPERT_MAGIC;                          // emrin tanıtıcı değeri
//--- emri gönder
   if(!OrderSend(request,result))
      PrintFormat("OrderSend hatası %d",GetLastError());     // emir gönderilemiyorsa
//--- işlem bilgileri
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Pozisyon kapama için TRADE_ACTION_DEAL işleminin bir örneği:

```
#define EXPERT_MAGIC 123456   // uzmanın tanıtıcı değeri (MagicNumber)
//+------------------------------------------------------------------+
//| Tüm pozistonların kapatılması                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- alım-satım isteğininin ve istek sonucunun bildirimini yap ve başlat
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // açık pozisyonların sayısı
//--- tüm pozisyonar için tekrarla
   for(int i=total-1; i>=0; i--)
     {
      //--- emir parametreleri
      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);
      //--- pozisyonnun çıktı verileri
      PrintFormat("#%I64u %s  %s  %.2f  %s [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  magic);
      //--- tanıtıcı değer (MagicNumber) eşleşiyorsa
      if(magic==EXPERT_MAGIC)
        {
         //--- isteğin ve sonuç değerlerinin sıfırlanması
         ZeroMemory(request);
         ZeroMemory(result);
         //--- işlem parametrelerinin ayarlanması
         request.action   =TRADE_ACTION_DEAL;        // alım-satım işleminin türü
         request.position =position_ticket;          // pozisyonun fişi
         request.symbol   =position_symbol;          // sembol
         request.volume   =volume;                   // pozisyon hacmi
         request.deviation=5;                        // izin verilen slipaj miktarı
         request.magic    =EXPERT_MAGIC;             // pozisyonun tanıtıcı değeri (Ma
         //--- emir türünü ve fiyatı pozisyona göre ayarla
         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;
           }
         //--- kapanışa dair veriler
         PrintFormat("Close #%I64d %s %s",position_ticket,position_symbol,EnumToStrin
         //--- isteği gönder
         if(!OrderSend(request,result))
            PrintFormat("OrderSend hatası %d",GetLastError());  // istek gönderilemez
         //--- işlem verileri
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //---
        }
```

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

Bekleyen emir girmek için kullanılan TRADE_ACTION_PENDING işleminin bir örneği:

```
#property description "Bekleyen emir girme örneği"
#property script_show_inputs
#define EXPERT_MAGIC 123456                                  // Uzmanın tanıtıcı değeri (Ma
input ENUM_ORDER_TYPE orderType=ORDER_TYPE_BUY_LIMIT;   // emir tipi
//+------------------------------------------------------------------+
//| Bekleyen emirlerin girilmesi                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- alım-satım isteğininin ve istek sonucunun bildirimini yap ve başlat
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- bekleyen emirde kullanılacak parametreler
   request.action   =TRADE_ACTION_PENDING;                              // alım-satım
   request.symbol   =Symbol();                                          // sembol
   request.volume   =0.1;                                               // 0.1 lotluk
   request.deviation=2;                                                 // izin veriler
   request.magic    =EXPERT_MAGIC;                                      // emri giren
   int offset = 50;                                                     // puan cinsin
   double price;                                                        // emrin tetik
   double point=SymbolInfoDouble(_Symbol,SYMBOL_POINT);                 // puan değeri
   int digits=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);                 // basamak say
   //--- işlem tipinin denetimi
   if(orderType==ORDER_TYPE_BUY_LIMIT)
     {
      request.type     =ORDER_TYPE_BUY_LIMIT;                          // emir tipi
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;        // açılış fiya
      request.price    =NormalizeDouble(price,digits);                 // normalleşti
     }
   else if(orderType==ORDER_TYPE_SELL_LIMIT)
     {
      request.type     =ORDER_TYPE_SELL_LIMIT;                          // emir tipi
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point;         // açılış fiya
      request.price    =NormalizeDouble(price,digits);                 // normalleşti
     }
   else if(orderType==ORDER_TYPE_BUY_STOP)
     {
      request.type =ORDER_TYPE_BUY_STOP;                               // emir tipi
      price        =SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point; // açılış fiya
      request.price=NormalizeDouble(price,digits);                     // normalleşti
     }
   else if(orderType==ORDER_TYPE_SELL_STOP)
     {
      request.type     =ORDER_TYPE_SELL_STOP;                          // emir tipi
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;        // açılış fiya
      request.price    =NormalizeDouble(price,digits);                 // normalleşti
     }
   else Alert("bu örnek sadece bekleyen emirler için geçerlidir");   // değilse, bekle
//--- isteği gönder
   if(!OrderSend(request,result))
      PrintFormat("OrderSend hatası %d",GetLastError());                // istek gönd
//--- işlemle ilgili veriler
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Bir açık pozisyonun Kr Al ve Zarar Durdur seviyelerini değiştirmek için TRADE_ACTION_SLTP işleminin örneği:

```
#define EXPERT_MAGIC 123456  // uzmanın tanıtıcı değeri (MagicNumber)
//+------------------------------------------------------------------+
//| Pozisyonun Kar Al ve Zarar Durdur seviyelerinin değişimi          |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- alım-satım isteğininin ve istek sonucunun bildirimini yap ve başlat
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // açık pozisyonların sayısı
//--- tüm pozisyonar için tekrarla
   for(int i=0; i<total; i++)
     {
      //--- emir parametreleri
      ulong  position_ticket=PositionGetTicket(i);// pozisyonun fişi
      string position_symbol=PositionGetString(POSITION_SYMBOL); // sembol
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS); // basamak
      ulong  magic=PositionGetInteger(POSITION_MAGIC); // Pozisyonun tanıtıcı değeri
      double volume=PositionGetDouble(POSITION_VOLUME);    // pozisyon hacmi
      double sl=PositionGetDouble(POSITION_SL);  // pozisyonun Zarar Durdur seviyesi
      double tp=PositionGetDouble(POSITION_TP);  // pozisyonun Kar Al seviyesi
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- pozisyonla ilgili veriler
      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);
      //--- tanıtıcı değer (MagicNumber) eşleşiyorsa Kar Al ve Zarar Durdur seviyeleri
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
```

```
         //--- mevcut fiyat seviyelerini hesapla
         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;
         //--- mevcut kapanış fiyatına göre izin verilen minium sapma değeri ayarlanma
         if(stop_level<=0)
            stop_level=150; // sapma değerini mevcut kapanışa göre 150 puan uzağa ayar
         else
            stop_level+=50; // güvenilirlik için uzaklığı (SYMBOL_TRADE_STOPS_LEVEL +

         //--- Zarar Durdur ve Kar Al seviye değerlerinin yuvarlanması
         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);
           }
         //--- istek ve sonuç değerlerini sıfırla
         ZeroMemory(request);
         ZeroMemory(result);
         //--- işlem parametrelerinin ayarlanması
         request.action  =TRADE_ACTION_SLTP; // alım-satım işleminin tipi
         request.position=position_ticket;   // pozisyonun fişi
         request.symbol=position_symbol;     // sembol
         request.sl      =sl;                // pozisyonun Zarar Durdur seviyesi
         request.tp      =tp;                // pozisyonun Kar Al seviyesi
         request.magic=EXPERT_MAGIC;         // Pozisyonun tanıtıcı değeri (MagicNumbe
         //--- değiştirme işlemine dair veriler
         PrintFormat("Değiştir #%I64d %s %s",position_ticket,position_symbol,EnumToStr
         //--- isteği gönder
         if(!OrderSend(request,result))
            PrintFormat("OrderSend hatası %d",GetLastError());  // istek gönderilemyo
         //--- işlemle ilgili veriler
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
//+------------------------------------------------------------------+
```

Bekleyen emirlerin fiyat seviyelerini değiştirmek için bir TRADE_ACTION_MODIFY örneği:

```
#define EXPERT_MAGIC 123456  // uzmanın tanıtıcı değeri (MagicNumber)
//+------------------------------------------------------------------+
//| Bekleyen emirlerin değiştirilmesi                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- alım-satım isteğininin ve istek sonucunun bildirimini yap ve başlat
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // bekleyen emirlerin toplam sayısı
//--- tüm bekleyen emirler için tekrarla
   for(int i=0; i<total; i++)
     {
      //--- emir parametreleri
      ulong  order_ticket=OrderGetTicket(i);                          // emir fişi
      string order_symbol=Symbol();                                   // sembol
      int    digits=(int)SymbolInfoInteger(order_symbol,SYMBOL_DIGITS); // basamak sa
      ulong  magic=OrderGetInteger(ORDER_MAGIC);                      // emri giren
      double volume=OrderGetDouble(ORDER_VOLUME_CURRENT);             // emrin mevc
      double sl=OrderGetDouble(ORDER_SL);                             // emrin mevc
      double tp=OrderGetDouble(ORDER_TP);                             // emrin mevc
      ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE); // emir tipi
      int offset = 50;                                                // puan cinsi
      double price;                                                   // emrin teti
      double point=SymbolInfoDouble(order_symbol,SYMBOL_POINT);       // puan değer
      //--- emirle ilgili çıktı verileri
      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);
      //---tanıtıcı değer (MagicNumber) eşleşiyorsa Kar Al ve Zarar Durdur seviyeleri
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
         request.action=TRADE_ACTION_MODIFY;                          // alım-satım
         request.order = OrderGetTicket(i);                           // emir fişi
         request.symbol   =Symbol();                                  // sembol
         request.deviation=5;                                         // izin veriler
        //--- fiyat seviyesinin, Zarar Durur ve Kar Al seviyelerinin emrin türüne göre
         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);              // normali
           }
         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);               // normal
           }
         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);                  // normal
           }
         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);                  // normal
           }
         //--- isteği gönder
         if(!OrderSend(request,result))
            PrintFormat("OrderSend hatası %d",GetLastError());  // istek gönderilemyo
         //--- işlemle ilgili veriler
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //--- istek ve sonuç değerlerini sıfırla
         ZeroMemory(request);
         ZeroMemory(result);
        }
     }
  }
//+------------------------------------------------------------------+
```

Bekleyen emirlerin silinmesi için bir TRADE_ACTION_REMOVE örneği:

```
#define EXPERT_MAGIC 123456  // uzmanın tanıtıcı değeri (MagicNumber)
//+------------------------------------------------------------------+
//| Bekleyen emirlerin silinmesi                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- alım-satım isteğininin ve istek sonucunun bildirimini yap ve başlat
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // bekleyen emirlerin toplam sayısı
//--- tüm bekleyen emirler için tekrarla
   for(int i=total-1; i>=0; i--)
     {
      ulong  order_ticket=OrderGetTicket(i);                   // emir fişi
      ulong  magic=OrderGetInteger(ORDER_MAGIC);               // emri giren uzmanın t
      //--- tanıtıcı değer (MagicNumber) eşleşiyorsa
      if(magic==EXPERT_MAGIC)
        {
         //--- istek ve sonuç değerlerini sıfırla
         ZeroMemory(request);
         ZeroMemory(result);
         //--- işlem parametrelerinin ayarlanması
         request.action=TRADE_ACTION_REMOVE;                   // alım-satım işleminir
         request.order = order_ticket;                         // emir fişi
         //--- isteği gönder
         if(!OrderSend(request,result))
            PrintFormat("OrderSend hatası %d",GetLastError());  // istek gönderilemyo
         //--- işlemle ilgili veriler
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
//+------------------------------------------------------------------+
```

Pozisyonları ters pozisyonlarla kapamak için bir TRADE_ACTION_CLOSE_BY örneği:

```
#define EXPERT_MAGIC 123456  // uzmanın tanıtıcı değeri (MagicNumber)
//+------------------------------------------------------------------+
//| Tüm pozisyonları ters pozisyonlarla kapat                        |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- alım-satım isteğininin ve istek sonucunun bildirimini yap ve başlat
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // açık pozisyonların sayısı
//--- tüm pozisyonar için tekrarla
   for(int i=total-1; i>=0; i--)
     {
      //--- emir parametreleri
      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);
      //--- pozisyonla ilgili veriler
      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);
      //--- tanıtıcı değer (MagicNumber) eşleşiyorsa
      if(magic==EXPERT_MAGIC)
        {
         for(int j=0; j<i; j++)
           {
            string symbol=PositionGetSymbol(j); // ters pozisyonun sembolü
            //--- düz ve ters poisyonların sembolleri uyuşuyorsa
            if(symbol==position_symbol && PositionGetInteger(POSITION_MAGIC)==EXPERT_M
              {
               //--- ters pozisyonun tipini ayarla
               ENUM_POSITION_TYPE type_by=(ENUM_POSITION_TYPE)PositionGetInteger(POSIT
               //--- düz ve ters sembollerin tipleri uyuşuyorsa bırak
               if(type==type_by)
                  continue;
               //--- istek ve sonuç değerlerini sıfırla
               ZeroMemory(request);
               ZeroMemory(result);
               //--- işlem parametrelerinin ayarlanması
               request.action=TRADE_ACTION_CLOSE_BY;                    // alım-s
               request.position=position_ticket;                        // pozisy
               request.position_by=PositionGetInteger(POSITION_TICKET); // ters p
               //request.symbol     =position_symbol;
               request.magic=EXPERT_MAGIC;                              // pozisy
               //--- ters pozisyonla kapama işlemine dair veriler
               PrintFormat("Close #%I64d %s %s by #%I64d",position_ticket,position_sym
               //--- isteği gönder
               if(!OrderSend(request,result))
                  PrintFormat("OrderSend hatası %d",GetLastError()); // istek gönderil
```

```
                //--- işlemle ilgili veriler
                PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result
               }
            }
         }
      }
   }
//+------------------------------------------------------------------+
```

Alım-Satım Faaliyet Tipleri

Alım-satım hesabı üzerinde bazı kesin işlemler gerçekleştirildiği zaman hesabın durumu değişir. Bu eylemler şunları kapsamaktadır:

- OrderSend ve OrderSendAsync fonksiyonları kullanılarak herhangi bir MQL5 uygulamasından yapılan alım-satım istekleri ve bunların ilerideki kullanımları;
- Terminalin grafiksel arayüzü ile yapılan alım satım istekleri ve bunların ilerideki kullanımları;
- Sunucu üzerindeki bekleyen emir ve durdurma emri aktivasyonu;
- İşlemlerin alım-satım sunucusu tarafından gerçekleştirilmesi.

Yukarıda sayılanlar eylemlerin sonucunda şu alım-satım faaliyetleri gerçekleşir:

- alım-satım isteğinin işlenmesi;
- açık emirlerin değişimi;
- emir geçmişinin değişimi;
- işlem geçmişinin değişimi;
- pozisyonların değişimi.

Örneğin, bir piyasa alım emri gönderirken, bu emir önce işlenir, hesap için uygun bir alım emri oluşturulur, sonra uygulanır ve açık emirler listesinden kaldırılır, ardından emir geçmişine eklenir, uygun bir işlem geçmişe eklenir ve yeni bir pozisyon oluşturulur. Tüm bu eylemler alım-satım faaliyetidir.

Programcının bir alım-satım hesabında gerçekleşmiş faaliyetleri takip edebilmesi için, OnTradeTransaction fonksiyonu tasarlanmıştır. Bu işleyici, bir hesap üzerinde uygulanmış alım-satım faaliyetlerinin MQL5 uygulaması içinde alınmasını sağlar. Alım-satım faaliyetinin tarifi, OnTradeTransaction fonksiyonunun ilk parametresinde, MqlTradeTransaction yapısı kullanılarak ibraz edilir.

Alım-satım faaliyetinin tipi, MqlTradeTransaction yapısının tip parametresinde belirtilir. Muhtemel alım-satım faaliyeti tipleri şu sayım ile tarif edilmiştir:

ENUM_TRADE_TRANSACTION_TYPE

Tanımlayıcı	Açıklama
TRADE_TRANSACTION_ORDER_ADD	Yeni bir açık emir ekleme.
TRADE_TRANSACTION_ORDER_UPDATE	Açılmış bir emri güncelle. Güncellemeler, sadece müşteri terminali veya alım-satım sunucusu tarafından yapılmış bariz değişimleri değil, aynı zamanda, emrin ayarlanması sırasında oluşan değişimleri de içerir (örneğin, ORDER_STATE_STARTED'dan ORDER_STATE_PLACED'a veya ORDER_STATE_PLACED'dan ORDER_STATE_PARTIAL'a vb. yapılan aktarımlar).
TRADE_TRANSACTION_ORDER_DELETE	Açık emirler listesinden bir emir silme. Bir emir, uygun bir isteğin ayarlanmasının veya uygulanmasının (karşılanmasının) ve geçmişe konulmasının bir sonucu olarak, açık emirler listesinden silinebilir.

Tanımlayıcı	Açıklama
TRADE_TRANSACTION_DEAL_ADD	Geçmişe bir işlem eklenmesi. Bu eylem, bir emrin işlenmesinin veya hesap bakiyesi ile işlemler gerçekleştirilmesinin bir sonucu olarak ortaya çıkar.
TRADE_TRANSACTION_DEAL_UPDATE	Geçmişteki bir işlemin güncellenmesi. Daha önce gerçekleştirilen bir işlemin, sunucu üzerinde değişme olasılığı bulunmaktadır. Örneğin, işlem daha önce aracı kurum tarafından aktarıldığı bir dışsal alım-satım sisteminde (borsa) değiştirilmiş olabilir.
TRADE_TRANSACTION_DEAL_DELETE	Geçmişteki bir işlemin silinmesi. Daha önce gerçekleştirilen bir işlemin sunucu üzerinde silinme olasılığı bulunmaktadır. Örneğin, işlem daha önce aracı kurum tarafından aktarıldığı bir dışsal alım-satım sisteminde (borsa) silinmiş olabilir.
TRADE_TRANSACTION_HISTORY_ADD	Bir işlemin veya iptalin sonucu olarak bir emrin geçmişe eklenmesi.
TRADE_TRANSACTION_HISTORY_UPDATE	Emir geçmişinde yer alan bir emrin değişmesi. Bu tip, sunucu tarafındaki fonksiyonelliğin artırılması için tasarlanmıştır.
TRADE_TRANSACTION_HISTORY_DELETE	Emir geçmişinden bir emrin silinmesi. Bu tip, sunucu tarafındaki fonksiyonelliğin artırılması için tasarlanmıştır.
TRADE_TRANSACTION_POSITION	İşlemle ilgili olmayan bir pozisyonun değiştirilmesi. Bu tür faaliyetler, bir pozisyonun sunucu tarafından değiştirildiğini gösterir. Pozisyon hacmi, açılış fiyatı, Zarar Durdur ve Kar Al seviyeleri değiştirilebilir. Değişimlere dair veriler, OnTradeTransaction işleyicisi ile MqlTradeTransaction yapısında belirtilir. İşlemin sonucunda pozisyon değişimi (ekleme, değiştirme veya kapama) olması, TRADE_TRANSACTION_POSITION faaliyetinin gerçekleşmesine yol açmaz.
TRADE_TRANSACTION_REQUEST	Bir alım-satım isteğinin sunucu tarafından işlendiğine ve işlenen sonucunun alındığına dair uyarı. Bunun gibi faaliyetler için sadece MqlTradeTransaction yapısının "type" alanı incelenmelidir. OnTradeTransaction fonksiyonunun ikinci ve üçüncü parametreleri (request ve result), ek veriler elde etmek için incelenmelidir.

Bir alım-satım faaliyetinin tipine bağlı olarak, MqlTradeTransaction yapısı içinde bu faaliyeti tanımlayan çeşitli parametreler doldurulur. İbraz edilen verilerin detaylı açıklamaları, "Bir Alım-Satım Faaliyetinin Yapısı" başlığı altında gösterilmiştir.

Ayrıca Bakınız

Bir Alım-Satım Faaliyetinin Yapısı, OnTradeTransaction

Piyasa Derinliğinde Alım-Satım Emirleri

Piyasa Derinliği penceresi, hisse senetleri için mevcuttur. Burada mevcut alım-satım emirlerini görebilirsiniz. Bir alım-satım işleminde arzu edilen yön, istenen miktar ve fiyat, her emir için belirtilir.

MQL5 araçları ile Piyasa Derinliğinin (PD) durumu hakkında bilgi almak amacıyla, MarketBookGet() fonksiyonu kullanılır. Bu fonksiyon, PD "ekran görüntüsünü" MqlBookInfo yapı dizisinin içine yerleştirir. Dizinin her elemanı *type* alanında emir yönü ile ilgili bilgi içerir - ENUM_BOOK_TYPE sayımının değeri.

ENUM_BOOK_TYPE

Tanımlayıcı	Açıklama
BOOK_TYPE_SELL	Satış Emri
BOOK_TYPE_BUY	Alış Emri
BOOK_TYPE_SELL_MARKET	Piyasa fiyatından satış emri
BOOK_TYPE_BUY_MARKET	Piyasa fiyatından alış emri

Ayrıca Bakınız

Yapılar ve sınıflar, PD Yapısı, Alım-satım işlem tipleri, Piyasa Bilgisi

Sinyal Özellikleri

Aşağıdaki sayımlar alım-satım sinyalleriyle ve sinyal kopyalama ayarlarıyla çalışmak için kullanılır.

Alım-satım sinyallerinin double tipli özelliklerinin sayımı:

ENUM_SIGNAL_BASE_DOUBLE

Tanıtıcı	Açıklama
SIGNAL_BASE_BALANCE	Hesap bakiyesi
SIGNAL_BASE_EQUITY	Hesap varlıkları
SIGNAL_BASE_GAIN	Hesap kazancı
SIGNAL_BASE_MAX_DRAWDOWN	Hesabın maksimum düşüş bilgisi
SIGNAL_BASE_PRICE	Sinyalin abonelik fiyatı
SIGNAL_BASE_ROI	Yatırım getirisi (%)

Alım-satım sinyallerinin tamsayı tipli özelliklerinin sayımı:

ENUM_SIGNAL_BASE_INTEGER

Tanıtıcı	Açıklama
SIGNAL_BASE_DATE_PUBLISHED	Yayınlanma tarihi (aboneler için piyasaya sürüldüğü tarih)
SIGNAL_BASE_DATE_STARTED	İzleme başlangıç tarihi
SIGNAL_BASE_DATE_UPDATED	Sinyalin işlem istatistiklerinin son güncelleme tarihi
SIGNAL_BASE_ID	Sinyalin Tanıtıcısı
SIGNAL_BASE_LEVERAGE	Hesap kaldıracı
SIGNAL_BASE_PIPS	Pip cinsinden kar
SIGNAL_BASE_RATING	Sinyalin derecelendirme durumu
SIGNAL_BASE_SUBSCRIBERS	Abone sayısı
SIGNAL_BASE_TRADES	İşlem sayısı
SIGNAL_BASE_TRADE_MODE	Hesap tipi (0-gerçek, 1-deneme, 2-yarışma)

Alım-satım sinyallerinin string tipli özelliklerinin sayımı:

ENUM_SIGNAL_BASE_STRING

Tanıtıcı	Açıklama
SIGNAL_BASE_AUTHOR_LOGIN	Yazar bilgisi
SIGNAL_BASE_BROKER	Aracı kurum ismi

Tanıtıcı	Açıklama
SIGNAL_BASE_BROKER_SERVER	Aracı kurum sunucusu
SIGNAL_BASE_NAME	Sinyal ismi
SIGNAL_BASE_CURRENCY	Sinyalin baz döviz cinsi

Alım-satım sinyali ayarlarının double tipli özelliklerinin sayımı:

ENUM_SIGNAL_INFO_DOUBLE

Tanıtıcı	Açıklama
SIGNAL_INFO_EQUITY_LIMIT	Varlık limiti
SIGNAL_INFO_SLIPPAGE	Slipaj (eşitleme sırasında piyasa emirleri verildiğinde veya işlem kopyalarken kullanılır)
SIGNAL_INFO_VOLUME_PERCENT	Maksimum mevduat oranı (%), r/o

Alım-satım sinyali ayarlarının tamsayı tipli özelliklerinin sayımı:

ENUM_SIGNAL_INFO_INTEGER

Tanıtıcı	Açıklama
SIGNAL_INFO_CONFIRMATIONS_DISABLED	Doğrulama beklemeden eşitlemeye başlamak için kullanılan bayrak
SIGNAL_INFO_COPY_SLTP	Kar Al ve Zarar Durdur emirleri için kopyalama bayrağı
SIGNAL_INFO_DEPOSIT_PERCENT	Mevduat yüzdesi (%)
SIGNAL_INFO_ID	Sinyalin tanıtıcısı, r/o
SIGNAL_INFO_SUBSCRIPTION_ENABLED	"Üyelikten sonra kopyalamaya başla" izninin bayrağı
SIGNAL_INFO_TERMS_AGREE	"Sinyaller hizmeti kullanım şartları" için kabul bayrağı, r/o

Alım-satım sinyali ayarlarının string tipli özelliklerinin sayımı:

ENUM_SIGNAL_INFO_STRING

Tanıtıcı	Açıklama
SIGNAL_INFO_NAME	Sinyal ismi, r/o

Ayrıca bakınız

Alım-satım sinyalleri

İsimlendirilmiş Sabitler

MQL5 dilinde kullanılan tüm sabitler, şu gruplar bölünebilir:

- Ön tanımlı makro ikameleri - derleme sırasında ikame edilen değerler;
- Matematiksel sabitler - bazı matematiksel ifadelerin değerleri;
- Nümerik tip sabitleri - basit tip kısıtlamalarının bazıları;
- Sonlandırma sebebi kodu - Sonlandırma sebeplerinin açıklaması;
- Nesne İşaretçisinin Kontrolü - CheckPointer() fonksiyonu ile dönüş yapılan işaretçi tipleri sayımı;
- Diğer sabitler - tüm diğer sabitler.

Öntanımlı Makro İkameleri

Hata ayıklama sürecini basitleştirmek ve MQL5 programı hakkında bilgi edinmek amacıyla, değerleri derleme sırasında belirlenen özel makro sabitleri bulunmaktadır. Bu sabitleri kullanmanın en basit yolu, aşağıdaki örnekten de görülebileceği gibi, Print() fonksiyonu ile değerleri çıktılamaktır.

Sabit	Açıklama
__CPU_ARCHITECTURE__	EX5 dosyasının derlendiği mimarinin (komut kümesi) adı
__DATE__	Dosya derleme tarihi (zaman olmadan, yani saatler, dakikalar ve saniyeler 0'a eşit olacak şekilde)
__DATETIME__	Dosya derleme tarihi ve zamanı
__LINE__	Kaynak kodunda makronun yer aldığı satır sayısı
__FILE__	Derlenmiş mevcut dosyanın ismi
__PATH__	Derlenmekte olan dosyanın, salt adresi
__FUNCTION__	Makronun, gövdesine yerleştirildiği fonksiyonun ismi
__FUNCSIG__	Makro gövdesinin yerleştirildiği fonksiyonun işareti. aşırı-yüklenmiş fonksiyonların tanımlanmasında, fonksiyonun eksiksiz bir tarifi kullanışlı olabilir
__MQLBUILD__, __MQL5BUILD__	Derleyici sürüm numarası
__COUNTER__	Karşılaşılan her __COUNTER__ bildirimi için derleyici, sayaç değerini 0 ile N-1 arasında değiştirir; burada N, koddaki kullanım sayısıdır. Kaynak kodu yeniden derlenirken __COUNTER__ sırası değişiklik yapılmadan garanti edilir. __COUNTER__ değeri aşağıdaki şekilde hesaplanır: • ilk sayaç değeri 0'dır, • sayacın her kullanımından sonra değeri 1 artar, • ilk olarak, derleyici tüm makroları ve şablonları kaynak kodunda yerlerinde genişletir, • her bir şablon fonksiyonu için ayrı bir kod oluşturulur, • her bir şablon sınıfı/yapısı için de ayrı bir kod oluşturulur, • daha sonra, derleyici elde edilen kaynak kodun üzerinden tanımlanan sırayla geçer ve tespit edilen her __COUNTER__ kullanımını geçerli sayaç değeriyle değiştirir. Aşağıdaki örnek, derleyicinin kaynak kodunu nasıl işlediğini ve karşılaştığı __COUNTER__ örneklerini sırayla artan değerlerle nasıl değiştirdiğini göstermektedir.
__RANDOM__	Derleyici, her __RANDOM__ bildirimi için rastgele bir ulong tipi değer ekler.

Örnek:

```
#property copyright "Copyright © 2009, MetaQuotes Software Corp."
#property link      "https://www.metaquotes.net"
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- Uzman Danışman başlatılması sırasında örnek bilgi çıktısı
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//--- timer olayları arasındaki aralığı ayarla
   EventSetTimer(5);
//---
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Uzman danışman sonlandırılması sırasındaki bilgi çıktısı
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
//---
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- tik alındığı zamanki bilgi çıktısı
   Print(" __MQLBUILD__ = ",__MQLBUILD__,"  __FILE__ = ",__FILE__);
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
   test2();
//---
  }
//+------------------------------------------------------------------+
//| test1                                                            |
//+------------------------------------------------------------------+
void test1(string par)
  {
//--- fonksiyon içi bilgi çıktısı
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__," par=",par);
  }
//+------------------------------------------------------------------+
//| test2                                                            |
//+------------------------------------------------------------------+
void test2()
  {
```

```
//--- fonksiyon içi bilgi çıktısı
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
  }
//+------------------------------------------------------------------+
//| OnTimer event handler                                            |
//+------------------------------------------------------------------+
void OnTimer()
  {
//---
   Print(" __FUNCTION__ = ",__FUNCTION__,"  __LINE__ = ",__LINE__);
   test1(__FUNCTION__);
  }
```

__COUNTER__ makrosu ile nasıl çalışılacağını öğrenmek için örnek

```
//--- ifadenin ve değerinin günlükte hızlı bir şekilde görüntülenmesi için bir makro
#define print(expr) Print(#expr,"=",expr)

//--- MACRO_COUNTER özel makrosunu önceden tanımlanmış __COUNTER__ makrosu aracılığıyl
#define MACRO_COUNTER __COUNTER__

//--- __COUNTER__ makrosunu kullanarak değişkenin girdi değerini ayarla
input int InpVariable = __COUNTER__;

//--- fonksiyonları tanımlamadan önce __COUNTER__ makrosunu kullanarak global değişken
int ExtVariable = __COUNTER__;

//+------------------------------------------------------------------+
//| __COUNTER__ değerini geri döndüren fonksiyon                     |
//+------------------------------------------------------------------+
int GlobalFunc(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| __COUNTER__ değerini geri döndüren şablon fonksiyonu             |
//+------------------------------------------------------------------+
template<typename T>
int GlobalTemplateFunc(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| __COUNTER__ değerini geri döndüren metoda sahip yapı             |
//+------------------------------------------------------------------+
struct A
  {
   int               dummy;  // kullanılmamış
```

```
   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| __COUNTER__ değerini geri döndüren metoda sahip şablon yapısı    |
//+------------------------------------------------------------------+
template<typename T>
struct B
  {
   int               dummy;  // kullanılmamış

   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| __COUNTER__ değerini geri döndüren şablon metoduna sahip yapı    |
//+------------------------------------------------------------------+
struct C
  {
   int               dummy;  // kullanılmamış

   template<typename T>
   int               Method(void)
     {
      return(__COUNTER__);
     }
  };
//+------------------------------------------------------------------+
//| __COUNTER__ değerini geri döndüren fonksiyon #2                  |
//+------------------------------------------------------------------+
int GlobalFunc2(void)
  {
   return(__COUNTER__);
  }
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart(void)
  {
// makroda ve değişkenlerde __COUNTER__
   print(MACRO_COUNTER);
   print(InpVariable);
   print(ExtVariable);
```

```
//--- fonksiyonlarda __COUNTER__
   print(GlobalFunc());
   print(GlobalFunc());                  // değer değişmedi
   print(GlobalTemplateFunc<int>());
   print(GlobalTemplateFunc<int>());     // değer değişmedi
   print(GlobalTemplateFunc<double>());// değer değişti
   print(GlobalFunc2());
   print(GlobalFunc2());                 // değer değişmedi

// yapıda __COUNTER__
   A a1, a2;
   print(a1.Method());
   print(a2.Method());                   // değer değişmedi

// şablon yapısında __COUNTER__
   B<int> b1, b2;
   B<double> b3;
   print(b1.Method());
   print(b2.Method());                   // değer değişmedi
   print(b3.Method());                   // değer değişti

// şablon fonksiyonuna sahip yapıda __COUNTER__
   C c1, c2;
   print(c1.Method<int>());
   print(c1.Method<double>());           // değer değişti
   print(c2.Method<int>());              // ilk c1.Method<int>() çağrısıyla aynı değer

//--- makro ve global değişkendeki __COUNTER__'a tekrar bir göz atalım
   print(MACRO_COUNTER);  // değer değişti
   print(ExtGlobal2);
  }
//--- fonksiyonları tanımladıktan sonra __COUNTER__ makrosunu kullanarak global değiş
int ExtGlobal2 = __COUNTER__;
//+------------------------------------------------------------------+

/* Sonuç
   __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

*/
```

Matematiksel Sabitler

Değer içeren özel sabitler, bazı matematiksel ifadeler için rezerve edilmiştir. Bu sabitler, matematiksel fonksiyonların yerine, programın herhangi bir yerinde değerlerin hesaplanması için kullanılabilir.

Sabit	Açıklama	Değer
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	0.78539816339744830961 6
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

Örnek:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- sabitlerin değerlerini çıktıla
   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));
```

```
}
```

Nümerik Tip Sabitleri

Her bir basit sayısal tip belli başlı görev türleri için düşünülmüştür ve doğru kullanıldığında bir MQL5 program işleminin optimize edilmesini sağlar. Daha iyi bir kod okunurluğunun sağlanması ve hesaplama sonuçlarının daha düzgün işlenebilmesi amacıyla, belli bir tipteki basit verilerdeki ayarlanmış kısıtlamalar hakkında bilgi almayı sağlayan sabitler bulunmaktadır.

Sabit	Açıklama	Değer
CHAR_MIN	char tipi ile temsil edilebilecek minimal değer	-128
CHAR_MAX	char tipi ile temsil edilebilecek maksimal değer	127
UCHAR_MAX	uchar tipi ile temsil edilebilecek maksimal değer	255
SHORT_MIN	uchar tipi ile temsil edilebilecek minimal değer	-32768
SHORT_MAX	short tipi ile temsil edilebilecek maksimal değer	32767
USHORT_MAX	ushort tipi ile temsil edilebilecek maksimal değer	65535
INT_MIN	int tipi ile temsil edilebilecek minimal değer	-2147483648
INT_MAX	int tipi ile temsil edilebilecek maksimal değer	2147483647
UINT_MAX	uint tipi ile temsil edilebilecek maksimal değer	4294967295
LONG_MIN	long tipi ile temsil edilebilecek minimal değer	-9223372036854775808
LONG_MAX	long tipi ile temsil edilebilecek maksimal değer	9223372036854775807
ULONG_MAX	ulong tipi ile temsil edilebilecek maksimal değer	18446744073709551615
DBL_MIN	double tipi ile temsil edilebilecek minimal pozitif değer	2.2250738585072014e-308
DBL_MAX	double tipi ile temsil edilebilecek maksimal değer	1.7976931348623158e+308
DBL_EPSILON	Şu koşulu sağlayan minimal değer: 1.0+DBL_EPSILON != 1.0 (double tipi için)	2.2204460492503131e-016
DBL_DIG	double tipi için, anlamlı ondalık hanelerin sayısı	15

Sabit	Açıklama	Değer
DBL_MANT_DIG	double tipi için mantis içindeki bit sayısı	53
DBL_MAX_10_EXP	double tipli maksimal ondalık üs değeri	308
DBL_MAX_EXP	double tipli maksimal ikili üs değeri	1024
DBL_MIN_10_EXP	double tipli minimal ondalık üs değeri	(-307)
DBL_MIN_EXP	double tipli minimal ikili üs değeri	(-1021)
FLT_MIN	float tipiyle temsil edilebilecek minimal pozitif değer	1.175494351e-38
FLT_MAX	float tipiyle temsil edilebilecek maksimal değer	3.402823466e+38
FLT_EPSILON	Şu koşulu sağlayan minimal değer: 1.0+DBL_EPSILON != 1.0 (float tipi için)	1.192092896e-07
FLT_DIG	float tipi için anlamlı ondalık hanelerin sayısı	6
FLT_MANT_DIG	float tipi için mantis içindeki bit sayısı	24
FLT_MAX_10_EXP	float tipli maksimal ondalık üs değeri	38
FLT_MAX_EXP	float tipli maksimal ikili üs değeri	128
FLT_MIN_10_EXP	float tipli minimal ondalık üs değeri	-37
FLT_MIN_EXP	float tipli minimal ikili üs değeri	(-125)

Örnek:

```
void OnStart()
  {
//--- sabit değerleri çıktıla
   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
*/
  }
```

Sonlandırma Sebebi Kodları

Sonlandırma sebebi kodlarına UninitializeReason() fonksiyonu ile dönüş yapılır. Muhtemel dönüş değerleri şunlardır:

Sabit	Değer	Açıklama
REASON_PROGRAM	0	Uzman Danışmanın kendi işleminiExpertRemove() fonksiyon çağrısıyla sonlandırmış
REASON_REMOVE	1	Program çizelgeden silinmiş
REASON_RECOMPILE	2	Program yeniden derlenmiş
REASON_CHARTCHANGE	3	Sembol veya çizelge periyodu değişmiş
REASON_CHARTCLOSE	4	Çizelge kapatılmış
REASON_PARAMETERS	5	Giriş parametreleri kullanıcı tarafından değiştirilmiş
REASON_ACCOUNT	6	Yeni bir hesap aktive edilmiş veya hesap ayarlarındaki bir değişiklik nedeniyle alım-satım sunucusuna yeniden bağlanılmış
REASON_TEMPLATE	7	Yeni bir şablon uygulanmış
REASON_INITFAILED	8	Bu değer, OnInit() işleyicisinin sıfır olmayan bir değere dönüş yaptığı anlamına gelir
REASON_CLOSE	9	Terminal kapatılmış

Sonlandırma sebebi kodu, aynı zamanda öntanımlı OnDeinit(const int reason) fonksiyonuna da parametre olarak geçirilir.

Örnek:

```
//+------------------------------------------------------------------+
//| Metin açıklamasını al                                            |
//+------------------------------------------------------------------+
string getUninitReasonText(int reasonCode)
  {
   string text="";
//---
   switch(reasonCode)
     {
      case REASON_ACCOUNT:
         text="Hesap değiştirildi";break;
      case REASON_CHARTCHANGE:
         text="Sembol veya zaman-aralığı değiştirildi";break;
      case REASON_CHARTCLOSE:
         text="Çizelge kapatıldı";break;
      case REASON_PARAMETERS:
         text="Giriş parametresi değiştirildi";break;
      case REASON_RECOMPILE:
```

```
         text="Program "+__FILE__+" yeniden derlendi";break;
      case REASON_REMOVE:
         text="Program "+__FILE__+" çizelgeden kaldırıldı";break;
      case REASON_TEMPLATE:
         text="Yeni şablon, çizelgeye uygulandı";break;
      default:text="Diğer bir neden";
     }
//---
   return text;
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Sonlandırma sebebi kodunu almanın ilk yolu
   Print(__FUNCTION__,"_Sonlandırma sebebi kodu = ",reason);
    //--- Sonlandırma sebebi kodunu almanın ikinci yolu
   Print(__FUNCTION__,"_UninitReason = ",getUninitReasonText(_UninitReason));
  }
```

Nesne İşaretçisinin Kontrolü

CheckPointer() fonksiyonu, nesne işaretçisinin tipini kontrol etmek amacıyla kullanılır. Fonksiyon, ENUM_POINTER_TYPE sayımının değerlerinden birine dönüş yapar. Eğer hatalı bir işaretçi kullanılmışsa, programın çalışması anında sonlandırılır.

new() operatörü ile oluşturulan nesneler, POINTER_DYNAMIC tipindedir. delete() operatörü, sadece bu tip işaretçiler için kullanılabilirdir ve kullanılmalıdır.

Tüm diğer işaretçiler POINTER_AUTOMATIC tipindedir. Bu, nesnenin MQL5 program ortamında otomatik olarak oluşturulduğu anlamına gelir. Böyle nesneler kullanıldıktan sonra otomatik olarak silinirler.

ENUM_POINTER_TYPE

Sabit	Açıklama
POINTER_INVALID	Hatalı işaretçi
POINTER_DYNAMIC	new() operatörü ile oluşturulmuş nesnenin işaretçisi
POINTER_AUTOMATIC	Otomatik olarak (new() kullanılmadan) oluşturulmuş herhangi bir nesnenin işaretçisi

Ayrıca Bakınız

Çalışma Zamanı Hataları, Nesne Silme Operatörü delete, CheckPointer

Diğer Sabitler

CLR_NONE sabiti rengin olmadığını belirtmek amacıyla kullanılır, yani göstergedeki grafiksel nesnenin veya grafiksel serilerin çizilmeyeceğini belirtir. Bu sabit, Web-renkleri sabitlerinin listesinde yer almaz ama renk argümanlarının gerektiği her yere uygulanabilir.

INVALID_HANDLE sabiti, dosya tanıtıcı değerlerini kontrol etme amaçlı kullanılabilir (bakınız FileOpen() ve FileFindFirst()).

Sabit	Açıklama	Değer
CHARTS_MAX	Terminalde aynı anda açık olabilecek maksimum çizelge sayısı	100
clrNONE	Rengin olmaması	-1
EMPTY_VALUE	Bir gösterge tamponu için boş değer	DBL_MAX
INVALID_HANDLE	Hatalı tanıtıcı değer	-1
IS_DEBUG_MODE	Bir MQ5 programının hata ayıklama modunda çalıştığını gösteren bayrak	Hata ayıklama modunda sıfır değilken, diğer durumlarda sıfırdır
IS_PROFILE_MODE	Bir MQ5 programının profilleme modunda çalıştığını gösteren bayrak	profilleme modunda sıfır değilken, diğer durumlarda sıfırdır
NULL	Tüm tipler için sıfırdır	0
WHOLE_ARRAY	Dizinin sonuna kadar var olan eleman sayısıdır, yani bütün dizinin işleneceğini belirtir	-1
WRONG_VALUE	Sabit, gizli (örtülü) olarak herhangi bir sayım tipine dönüştürülebilir	-1

EMPTY_VALUE sabiti, genellikle göstergenin çizelgede görüntülenmeyen değerlerine karşılık gelir. Örneğin kullanıma hazır göstergelerden, 20 periyot değerine sahip Standard Deviation (standart sapma) göstergesi için, geçmişteki ilk 19 çubukluk kısım çizelge üzerinde gösterilmez. Eğer iStdDev() ile bu göstergenin tanıtıcı değerini oluşturursanız ve tanıtıcı değeri söz konusu çubuklar için CopyBuffer() ile gösterge değerlerinden oluşan bir diziye kopyalarsanız, değerler EMPTY_VALUE olarak gözükecektir.

Herhangi bir özel gösterge için, çizelgeye çizilmeyecek değerleri boş değerler şeklinde ayarlayabilirsiniz. Bunun için PlotIndexSetDouble() fonksiyonunu PLOT_EMPTY_VALUE şekillendiricisi ile kullanın.

NULL sabiti, basit tipli tüm değişkenlere, nesne yapılarına veya sınıf işaretçilerine atanabilir. Bir dizgi değişkenine yapılan NULL ataması, bu değişkenin tamamen sonlandırılması anlamına gelir.

WRONG_VALUE sabiti, hatalı bir sayım değerine dönüş yapılması gereken durumlar için düşünülmüştür. Örneğin, bir dönüş değerinin, bir sayımın değerlerinden olup olmadığıyla ilgili bilgiye ihtiyacımız olduğunda bu sabiti kullanabiliriz. İsmiyle belirtilen bir nesne için çizgi stiline dönüş yapan CheckLineStyle() fonksiyonunu göz önüne alalım. Eğer ObjectGetInteger() ile yapılan stil kontrolünde sonuç 'true' ise, ENUM_LINE_STYLE sayımından bir değere; aksi durumda ise WRONG_VALUE değerine dönüş yapılır.

```
void OnStart()
  {
   if(CheckLineStyle("MyChartObject")==WRONG_VALUE)
      printf("Çizgi stilinin alınmasında hata.");
  }
//+------------------------------------------------------------------+
//| İsimle belirtilen bir nesnenin çizgi stiline dönüş yapar         |
//+------------------------------------------------------------------+
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);
  }
```

WHOLE_ARRAY sabiti, işlenen dizideki eleman sayısının belirtilmesini gerektiren fonksiyonlar için düşünülmüştür:

- ArrayCopy();
- ArrayMinimum();
- ArrayMaximum();
- FileReadArray();
- FileWriteArray().

Eğer belirtilen değerden son değere kadar tüm dizi elemanlarının işleneceğini belirtmek istiyorsanız, sadece WHOLE_ARRAY değerini belirtmeniz yeterli olacaktır.

IS_PROFILE_MODE sabiti, profilleme modunda düzgün veri toplama amacıyla bir program işlevinin değiştirmesini sağlar. Profilleme, (genellikle fonksiyonları içeren) her bir tekil program bölümünün uygulama zamanının ölçülmesini ve aynı zamanda çağrı sayısının hesaplanmasını sağlar. Aşağıdaki örnekteki gibi, Sleep() fonksiyonu çağrıları, profilleme modunda uygulama zamanının belirlenmesini devre dışı bırakabilir:

```
//--- Sleep, profilleme sonucunu büyük ölçüde değiştirebilir
if(!IS_PROFILE_MODE) Sleep(100); // profilleme modunda Sleep() çağrısını devre dışı b
```

IS_PROFILE_MODE sabit değeri, derleme sırasında derleyici tarafından ve genellikle sıfır olarak ayarlanır. Profilleme modunda bir programı çalıştırırken, özel bir derleme gerçekleştirilir ve IS_PROFILE_MODE değeri, sıfır olmayan bir değer ile değiştirilir.

IS_DEBUG_MODE sabiti, hata ayıklama modunda bir MQL5 programının işleyişini hafifçe değiştirmek istediğinizde kullanışlı olabilir. Örneğin, hata ayıklama modunda bazı ek bilgileri terminal günlüğünde görüntülemek veya bir çizelge içinde fazladan grafik nesneleri oluşturmak isteyebilirsiniz.

Aşağıdaki örnekte, bir Label (etiket) nesnesi oluşturulur ve bunun açıklaması ve rengi script uygulama moduna bağlı olarak ayarlanır. MetaEditor içinde bir betiği hata-ayıklama modunda çalıştırmak için, F5 tuşuna basın. Eğer betiği terminaldeki tarayıcı penceresinden çalıştırırsanız, Label nesnesinin metni ve rengi farklı olacaktır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                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"
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   string label_name="invisible_label";
   if(ObjectFind(0,label_name)<0)
     {
      Print("Nesne",label_name,"bulunamadı. Hata kodu = ",GetLastError());
      //--- Label nesnesini oluştur
      ObjectCreate(0,label_name,OBJ_LABEL,0,0,0);
      //--- X koordinatını ayarla
      ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,200);
      //--- Y koordinatını ayarla
      ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,300);
      ResetLastError();
      if(IS_DEBUG_MODE) // hata-ayıklama modu
        {
         //--- script uygulama modu için mesaj görüntüle
         ObjectSetString(0,label_name,OBJPROP_TEXT,"DEBUG MODE");
         //--- metin rengini kırmızı olarak ayarla
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,clrRed))
            Print("Renk ayarlanamıyor. Hata",GetLastError());
        }
      else              // işlem modu
        {
         ObjectSetString(0,label_name,OBJPROP_TEXT,"RELEASE MODE");
         //--- metin rengini görünmez olarak ayarla
         if(!ObjectSetInteger(0,label_name,OBJPROP_COLOR,CLR_NONE))
            Print("Renk ayarlanamıyor. Hata",GetLastError());
        }
```

```
      ChartRedraw();
      DebugBreak();    // eğer hata-ayıklama modundaysak, burada sonlandırma gerçekles
     }
  }
```

Crypt Methods

The ENUM_CRYPT_METHOD enumeration is used to specify the data tranformation method, used in CryptEncode() and CryptDecode() functions.

ENUM_CRYPT_METHOD

Sabit	Açıklama
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

Ayrıca Bakınız

DebugBreak, Çalıştırılan MQL5 programı özellikleri, CryptEncode(), CryptDecode()

Veri Yapıları

MQL5 Dili 12 ön tanımlı yapı sunmaktadır:

- MqlDateTime, tarih ve zaman ile çalışmak için düşünülmüştür;
- MqlParam, bir göstergenin tanıtıcı değeri oluşturulurken, IndicatorCreate() fonksiyonunu kullanarak giriş parametrelerini yollayabilir;
- MqlRates yapısı, tarihsel veriyi manüple etmek içi düşünülmüştür, fiyat, hacim ve makas ile ilgili bilgi içerir;
- MqlBookInfo, Piyasa Derinliği hakkında bilgi elde etmek için düşünülmüştür;
- MqlTradeRequest, alım-satım işlemleri için alım-satım isteği oluşturmak amacıyla kullanılır;
- MqlTradeCheckResultfonksiyonu, hazırlanan alım-satım isteğini gönderilmeden önce kontrol etmek için düşünülmüştür;
- MqlTradeResult, OrderSend() fonksiyonu ile gönderilen bir alım-satım isteğini içerir;
- MqlTradeTransaction, bir alım-satım faaliyetinin tarifini içerir;
- MqlTick, mevcut fiyatlar hakkındaki en çok istenen bilgilerin hızlı şekilde elde edilmesi için tasarlanmıştır.
- Ekonomik takvim yapıları, gerçek zamanlı olarak MetaTrader 5 platformuna gönderilen ekonomik takvim olayları hakkında veri elde etmek için kullanılır. Ekonomik takvim fonksiyonları, yeni olay raporları açıklandıktan hemen sonra makroekonomik parametrelerin analiz edilmesini sağlar, çünkü ilgili değerler gecikmeden doğrudan kaynaktan yayımlanır.

MqlDateTime

Tarih tipi yapı int tipinde sekiz alan içerir:

```
struct MqlDateTime
  {
   int year;            // Yıl
   int mon;             // Ay
   int day;             // Gün
   int hour;            // Saat
   int min;             // Dakikalar
   int sec;             // Saniyeler
   int day_of_week;     // Haftanın günleri (0-Pazar, 1-Pazartesi, ... ,6-Cumartesi)
   int day_of_year;     // Yılın gün numarası (1 Ocak sıfır sayı değerine atanmıştır)
  };
```

Not

Artık yıl için, yılın DAY_OF_YEAR olan gün sayısı; Mart ayından itibaren artık olmayan bir yıl için bir dizi ilgili günden farklı olacaktır.

Örnek:

```
void OnStart()
  {
//---
   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, yılın günü = %d",str1.day,str1.mon,
          str1.year,str1.day_of_year);
   printf("%02d.%02d.%4d, yılın günü = %d",str2.day,str2.mon,
          str2.year,str2.day_of_year);
  }
/*  Sonuç:
  01.03.2008, yılın günü = 60
  01.03.2009, yılın günü = 59
*/
```

Ayrıca Bakınız

TimeToStruct, Yapılar ve Sınıflar

Göstergelerin giriş parametrelerinin yapısı (MqlParam)

MqlParam yapısı, IndicatorCreate() fonksiyonunu kullanarak bir teknik gösterge tanıtıcı değeri oluştururken giriş parametrelerinin tedarik edilmesi için özel olarak tasarlanmıştır

```
struct MqlParam
  {
   ENUM_DATATYPE     type;                // giriş parametrelerinin tipi, ENUM_DAT
   long              integer_value;       // bir tamsayı tipini saklamak için, ala
   double            double_value;        // bir double tipini saklamak için, alan
   string            string_value;        // bir string tipini saklamak için, alan
  };
```

Bir göstergenin tüm giriş parametreleri MqlParam tipli bir dizi şeklinde aktarılır; bu dizinin her bir elemanının *tip* alanı, elman ile aktarılan verinin tipini belirtir. Gösterge değerleri, *tip* alanında belirtilen ENUM_DATATYPE sayımının değerine bağlı olarak, her bir eleman için uygun alanlara (*integer_value* içine, *double_value* içine veya *string_value* içine) yerleştirilmelidir.

Eğer IND_CUSTOM değeri, IndicatorCreate() fonksiyonuna gösterge tipi şeklinde üçüncü olarak geçirilmişse, giriş parametreleri dizisinin ilk elemanı, ENUM_DATATYPE sayımından TYPE_STRING değeri ile *tip* alanına sahip olmalıdır ve *string_value* alanı özel göstergenin ismini içermelidir

MqlRates

Bu yapı; fiyatlar, hacimler ve makas hakkında bilgi depolar.

```
struct MqlRates
  {
   datetime time;         // Periyot başlangıç zamanı
   double   open;         // Açılış fiyatı
   double   high;         // Periyodun en yüksek değeri
   double   low;          // Periyodun en küçük değeri
   double   close;        // Kapanış fiyatı
   long     tick_volume;  // Tik hacmi
   int      spread;       // Makas
   long     real_volume;  // Alım-satım hacmi
  };
```

Örnek:

```
void OnStart()
  {
   MqlRates rates[];
   int copied=CopyRates(NULL,0,0,100,rates);
   if(copied<=0)
      Print("Fiyat verisini kopyalama başarısız ",GetLastError());
   else Print("",ArraySize(rates)," sayıda çubuk kopyalandı");
  }
```

Ayrıca Bakınız

CopyRates, Zaman serilerine erişim

MqlBookInfo

Piyasa derinliği hakkında bilgi sağlar.

```
struct MqlBookInfo
  {
   ENUM_BOOK_TYPE    type;          // ENUM_BOOK_TYPE sayımından emir tipi
   double            price;         // Fiyat
   long              volume;        // Hacim
   double            volume_real;   // Daha yüksek doğruluklu hacim
  };
```

Not

MqlBookInfo yapısı ön tanımlıdır, bu nedenle bildirim ve tarif gerektirmez. Yapıyı kullanmak için, bu tipten bir değişkeni sadece bildirmeniz yeterlidir.

PD sadece bazı semboller için mevcuttur.

Örnek:

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print(Symbol(), "hakkında MarketBookInfo");
     }
   else
     {
      Print("Sembol için PD alınamadı",Symbol());
     }
```

Ayrıca Bakınız

MarketBookAdd, MarketBookRelease, MarketBookGet, PD içinde Alım-Satım Emirleri, Veri Tipleri

Alım-Satım İsteği Yapısı (MqlTradeRequest)

Emir yerleştirme işleminin yapılması için müşteri terminali ve alım-satım sunucusu arasındaki etkileşim, alım-satım istekleri kullanılarak gerçekleştirilir. Alım-satım isteği, MqlTradeRequest tipli ön tanımlı özel bir yapı ile temsil edilir. Bu yapı, alım-satım işlemlerinin gerçekleştirilmesi için tüm gerekli alanları içerir. İstek sürecinin sonucu MqlTradeResult tipi ile temsil edilir.

```
struct MqlTradeRequest
  {
   ENUM_TRADE_REQUEST_ACTIONS    action;          // Alım-satım işlemi tipi
   ulong                         magic;           // Uzman Danışman kimliği (magic nu
   ulong                         order;           // Emir fişi
   string                        symbol;          // Alım-satım sembolü
   double                        volume;          // Lot bazında işlem için istenen h
   double                        price;           // Fiyat
   double                        stoplimit;       // Emrin StopLimit seviyesi
   double                        sl;              // Emrin Stop Loss (kaybı durdur) s
   double                        tp;              // Emrin Take Profit (kar al) sevi
   ulong                         deviation;       // İstenen fiyattan maksimum olası
   ENUM_ORDER_TYPE               type;            // Emir tipi
   ENUM_ORDER_TYPE_FILLING       type_filling;    // Emir gerçekleştirme tipi
   ENUM_ORDER_TYPE_TIME          type_time;       // Emir zaman aşımı tipi
   datetime                      expiration;      // Emir zaman aşımı zamanı (ORDER_T
   string                        comment;         // Emir yorumu
   ulong                         position;        // Pozisyon fişi
   ulong                         position_by;     // Ters pozisyonun fişi
  };
```

Alanların açıklaması

Alan	Açıklama
action	Alım-satım işlem tipi. ENUM_TRADE_REQUEST_ACTIONS sayımının değerlerinden biri olabilir.
magic	Uzman danışman kimliği. Alım-satım emirlerinin analitik olarak işlenmesinin düzenlenmesini sağlar. Her bir Uzman Danışman bir alım-satım isteği gönderirken kendi benzersiz kimliğini ayarlayabilir.
order	Emir fişi. Bekleyen emirler göndermek için kullanılır.
symbol	Emir sembolü. Emir değişimi ve pozisyon kapatma işlemleri için gerekli değildir.
volume	Lot bazında istenen emir hacmi. İşlemin gerçek hacminin emir gerçekleştirme tipine bağlı olduğunu not edin.
price	Emrin uygulanması gereken seviyeye ulaşan fiyat. Gerçekleştirme tipi, TRADE_ACTION_DEAL tipinden "Market Execution" (SYMBOL_TRADE_EXECUTION_MARKET) olan, sembollerin piyasa emirleri; fiyat belirlemesi gerektirmez

Alan	Açıklama
stoplimit	fiyat, istenen *fiyat* değerine ulaştığında (bu koşul zorunludur), Limit bekleyen emrin işleme konulacağı fiyat değeri. O zamana kadar, bekleyen emirler işleme konulmaz.
sl	Olumsuz fiyat hareketi durumundaki Stop Loss (zararı durdur) fiyatı
tp	Olumlu fiyat hareketi durumundaki Take Profit (kar al) fiyatı
deviation	Puan bazında belirtilen maksimum fiyat sapması
type	Emir tipi. ENUM_ORDER_TYPE sayımının değerlerinden biri olabilir.
type_filling	Emir gerçekleştirme tipi. ENUM_ORDER_TYPE_FILLING sayımının değerlerinden biri olabilir.
type_time	Emir zaman aşımı tipi. ENUM_ORDER_TYPE_TIME sayımının değerlerinden biri olabilir.
expiration	Emir zaman aşımı zamanı (ORDER_TIME_SPECIFIED tipli emirler için)
yorum	Emir yorumu
position	Pozisyonun fişi. Bir pozisyon kapatılırken veya değiştirilirken pozisyonu tanımlamak için girilmelidir. Pozisyon hangi emrin sonucunda açılmışsa o emrin fişi pozisyonun fişiyle aynıdır.
position_by	Ters pozisyon fişi. Aynı sembol üzerindeki bir pozisyonu ters pozisyonla kapatmak için kullanılır.

Hedge'li sistemde bir pozisyonu değiştirirken veya kapatırken, pozisyonun fişini belirttiğinizden emin olun (MqlTradeRequest::poisition). Fiş numarası netleştirme sisteminde de belirtilebilir ama bu sistemde pozisyonlar sembol ismlerine göre tanımlanır.

Alım-satım işlemlerini gerçekleştirecek emirler yollamak için OrderSend() fonksiyonunun kullanılması gereklidir. Her bir alım-satım işlemi için zorunlu alanların belirtilmesi gereklidir; isteğe bağlı alanlar da ayrıca doldurulabilir. Bir alım-satım emrinin gönderilmesinde, yedi olası durum söz konusudur:

İstek İşlemi

İstek işlemi talep edilen fiyatlar üzerinden pozisyon açmaya yarayan bir alım-satım emridir. Şu 9 alanın belirtilmesini gerektirir:

- action
- symbol
- volume
- price
- sl
- tp
- deviation
- type
- type_filling

Ayrıca "magic" ve "comment" alanlarının da belirtilmesi mümkündür.

Anlık İşlem

Anlık işlem (cari fiyatlar ile alım-satım) modunda pozisyon açan bir alım-satım emridir. Şu 9 alanın belirtilmesini gerektirir:

- action
- symbol
- volume
- price
- sl
- tp
- deviation
- type
- type_filling

Ayrıca "magic" ve "comment" alanlarının da belirtilmesi mümkündür.

Piyasa İşlemi

Bu, Piyasa İşlemi modunda pozisyon açan bir alım-satım emridir. Şu 5 alanın belirtilmesini gerektirir:

- action
- symbol
- volume
- type
- type_filling

Ayrıca "magic" ve "comment" alanlarının da belirtilmesi mümkündür.

Borsa İşlemi

Borsa İşlemi modunda pozisyon açan bir alım-satım emridir. Şu 5 alanın belirtilmesini gerektirir:

- action
- symbol
- volume
- type
- type_filling

Ayrıca "magic" ve "comment" alanlarının da belirtilmesi mümkündür.

Alış pozisyonu açmak için TRADE_ACTION_DEAL işleminin bir örneği:

```
#define EXPERT_MAGIC 123456   // uzmanın tanıtıcı değeri (MagicNumber)
//+------------------------------------------------------------------+
//| Alış pozisyonu açılışı                                           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- alım-satım isteğininin ve istek sonucunun bildirimini yap ve başlat
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- istek parametreleri
   request.action   =TRADE_ACTION_DEAL;                     // alım-satım işleminin tü
   request.symbol   =Symbol();                              // sembol
   request.volume   =0.1;                                   // 0.1 lotluk hacim lot
   request.type     =ORDER_TYPE_BUY;                        // emir türü
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_ASK); // açılış fiyatı
   request.deviation=5;                                     // izin verilen slipaj mik
   request.magic    =EXPERT_MAGIC;                          // emrin tanıtıcı değeri
//--- emri gönder
   if(!OrderSend(request,result))
      PrintFormat("OrderSend hatası %d",GetLastError());     // emir gönderilemiyorsa
//--- işlem bilgileri
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Satış pozisyonu açmak için TRADE_ACTION_DEAL işleminin bir örneği:

```
#define EXPERT_MAGIC 123456   // uzmanın tanıtıcı değeri (MagicNumber)
//+------------------------------------------------------------------+
//| Satış pozisyonu açılışı                                          |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- alım-satım isteğininin ve istek sonucunun bildirimini yap ve başlat
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- istek parametreleri
   request.action   =TRADE_ACTION_DEAL;                     // alım-satım işleminin tü
   request.symbol   =Symbol();                              // sembol
   request.volume   =0.2;                                   // 0.2 lotluk hacim
   request.type     =ORDER_TYPE_SELL;                       // emir türü
   request.price    =SymbolInfoDouble(Symbol(),SYMBOL_BID); // açılış fiyatı
   request.deviation=5;                                     // izin verilen slipaj mik
   request.magic    =EXPERT_MAGIC;                          // emrin tanıtıcı değeri
//--- emri gönder
   if(!OrderSend(request,result))
      PrintFormat("OrderSend hatası %d",GetLastError());     // emir gönderilemiyorsa
//--- işlem bilgileri
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Pozisyon kapama için TRADE_ACTION_DEAL işleminin bir örneği:

```
#define EXPERT_MAGIC 123456   // uzmanın tanıtıcı değeri (MagicNumber)
//+------------------------------------------------------------------+
//| Tüm pozistonların kapatılması                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- alım-satım isteğininin ve istek sonucunun bildirimini yap ve başlat
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // açık pozisyonların sayısı
//--- tüm pozisyonar için tekrarla
   for(int i=total-1; i>=0; i--)
     {
      //--- emir parametreleri
      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);
      //--- pozisyonnun çıktı verileri
      PrintFormat("#%I64u %s  %s  %.2f  %s [%I64d]",
                  position_ticket,
                  position_symbol,
                  EnumToString(type),
                  volume,
                  DoubleToString(PositionGetDouble(POSITION_PRICE_OPEN),digits),
                  magic);
      //--- tanıtıcı değer (MagicNumber) eşleşiyorsa
      if(magic==EXPERT_MAGIC)
        {
         //--- isteğin ve sonuç değerlerinin sıfırlanması
         ZeroMemory(request);
         ZeroMemory(result);
         //--- işlem parametrelerinin ayarlanması
         request.action   =TRADE_ACTION_DEAL;        // alım-satım işleminin türü
         request.position =position_ticket;          // pozisyonun fişi
         request.symbol   =position_symbol;          // sembol
         request.volume   =volume;                   // pozisyon hacmi
         request.deviation=5;                        // izin verilen slipaj miktarı
         request.magic    =EXPERT_MAGIC;             // pozisyonun tanıtıcı değeri (Ma
         //--- emir türünü ve fiyatı pozisyona göre ayarla
         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;
           }
         //--- kapanışa dair veriler
         PrintFormat("Close #%I64d %s %s",position_ticket,position_symbol,EnumToStrin
         //--- isteği gönder
         if(!OrderSend(request,result))
            PrintFormat("OrderSend hatası %d",GetLastError());  // istek gönderilemez
         //--- işlem verileri
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //---
        }
```

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

SL & TP Seviyelerinin Değiştirilmesi

Kar Al (TP) ve/veya Zarar Durdur (SL) fiyat seviyelerini değiştirmek için alım-satım emirleri. Şu 4 alanın belirtilmesini gerektirir:

- action
- symbol
- sl
- tp
- position

Bir açık pozisyonun Kr Al ve Zarar Durdur seviyelerini değiştirmek için TRADE_ACTION_SLTP işleminin örneği:

```
#define EXPERT_MAGIC 123456  // uzmanın tanıtıcı değeri (MagicNumber)
//+------------------------------------------------------------------+
//| Pozisyonun Kar Al ve Zarar Durdur seviyelerinin değişimi          |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- alım-satım isteğininin ve istek sonucunun bildirimini yap ve başlat
   MqlTradeRequest request;
   MqlTradeResult  result;
   int total=PositionsTotal(); // açık pozisyonların sayısı
//--- tüm pozisyonar için tekrarla
   for(int i=0; i<total; i++)
     {
      //--- emir parametreleri
      ulong  position_ticket=PositionGetTicket(i);// pozisyonun fişi
      string position_symbol=PositionGetString(POSITION_SYMBOL); // sembol
      int    digits=(int)SymbolInfoInteger(position_symbol,SYMBOL_DIGITS); // basamak
      ulong  magic=PositionGetInteger(POSITION_MAGIC); // Pozisyonun tanıtıcı değeri
      double volume=PositionGetDouble(POSITION_VOLUME);    // pozisyon hacmi
      double sl=PositionGetDouble(POSITION_SL);  // pozisyonun Zarar Durdur seviyesi
      double tp=PositionGetDouble(POSITION_TP);  // pozisyonun Kar Al seviyesi
      ENUM_POSITION_TYPE type=(ENUM_POSITION_TYPE)PositionGetInteger(POSITION_TYPE);
      //--- pozisyonla ilgili veriler
      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);
      //--- tanıtıcı değer (MagicNumber) eşleşiyorsa Kar Al ve Zarar Durdur seviyeler
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
```

```
         //--- mevcut fiyat seviyelerini hesapla
         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;
         //--- mevcut kapanış fiyatına göre izin verilen minium sapma değeri ayarlanma
         if(stop_level<=0)
            stop_level=150; // sapma değerini mevcut kapanışa göre 150 puan uzağa ayar
         else
            stop_level+=50; // güvenilirlik için uzaklığı (SYMBOL_TRADE_STOPS_LEVEL +

         //--- Zarar Durdur ve Kar Al seviye değerlerinin yuvarlanması
         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);
           }
         //--- istek ve sonuç değerlerini sıfırla
         ZeroMemory(request);
         ZeroMemory(result);
         //--- işlem parametrelerinin ayarlanması
         request.action  =TRADE_ACTION_SLTP; // alım-satım işleminin tipi
         request.position=position_ticket;   // pozisyonun fişi
         request.symbol=position_symbol;     // sembol
         request.sl      =sl;                // pozisyonun Zarar Durdur seviyesi
         request.tp      =tp;                // pozisyonun Kar Al seviyesi
         request.magic=EXPERT_MAGIC;         // Pozisyonun tanıtıcı değeri (MagicNumbe
         //--- değiştirme işlemine dair veriler
         PrintFormat("Değiştir #%I64d %s %s",position_ticket,position_symbol,EnumToStr
         //--- isteği gönder
         if(!OrderSend(request,result))
            PrintFormat("OrderSend hatası %d",GetLastError());  // istek gönderilemyo
         //--- işlemle ilgili veriler
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
//+------------------------------------------------------------------+
```

Bekleyen Emir

Bekleyen emri işleme koymak için alım-satım emri. Şu 11 alanın belirtilmesini gerektirir:

- action
- symbol
- volume
- price
- stoplimit
- sl
- tp
- type

- type_filling
- type_time
- expiration

Ayrıca "magic" ve "comment" alanlarının da belirtilmesi mümkündür.

Bekleyen emir girmek için kullanılan TRADE_ACTION_PENDING işleminin bir örneği:

```
#property description "Bekleyen emir girme örneği"
#property script_show_inputs
#define EXPERT_MAGIC 123456                                  // Uzmanın tanıtıcı değeri (Ma
input ENUM_ORDER_TYPE orderType=ORDER_TYPE_BUY_LIMIT;   // emir tipi
//+------------------------------------------------------------------+
//| Bekleyen emirlerin girilmesi                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- alım-satım isteğininin ve istek sonucunun bildirimini yap ve başlat
   MqlTradeRequest request={};
   MqlTradeResult  result={};
//--- bekleyen emirde kullanılacak parametreler
   request.action   =TRADE_ACTION_PENDING;                           // alım-satım i
   request.symbol   =Symbol();                                       // sembol
   request.volume   =0.1;                                            // 0.1 lotluk h
   request.deviation=2;                                              // izin veriler
   request.magic    =EXPERT_MAGIC;                                   // emri giren u
   int offset = 50;                                                  // puan cinsinc
   double price;                                                     // emrin tetikl
   double point=SymbolInfoDouble(_Symbol,SYMBOL_POINT);              // puan değeri
   int digits=SymbolInfoInteger(_Symbol,SYMBOL_DIGITS);              // basamak sayı
   //--- işlem tipinin denetimi
   if(orderType==ORDER_TYPE_BUY_LIMIT)
     {
      request.type     =ORDER_TYPE_BUY_LIMIT;                       // emir tipi
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK)-offset*point;     // açılış fiyat
      request.price    =NormalizeDouble(price,digits);              // normalleştir
     }
   else if(orderType==ORDER_TYPE_SELL_LIMIT)
     {
      request.type     =ORDER_TYPE_SELL_LIMIT;                       // emir tipi
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID)+offset*point;      // açılış fiya
      request.price    =NormalizeDouble(price,digits);               // normalleşti
     }
   else if(orderType==ORDER_TYPE_BUY_STOP)
     {
      request.type =ORDER_TYPE_BUY_STOP;                            // emir tipi
      price        =SymbolInfoDouble(Symbol(),SYMBOL_ASK)+offset*point; // açılış fiya
      request.price=NormalizeDouble(price,digits);                  // normalleşti
     }
   else if(orderType==ORDER_TYPE_SELL_STOP)
     {
      request.type     =ORDER_TYPE_SELL_STOP;                       // emir tipi
      price=SymbolInfoDouble(Symbol(),SYMBOL_BID)-offset*point;     // açılış fiya
      request.price    =NormalizeDouble(price,digits);              // normalleşti
     }
   else Alert("bu örnek sadece bekleyen emirler için geçerlidir");   // değilse, bekle
//--- isteği gönder
   if(!OrderSend(request,result))
      PrintFormat("OrderSend hatası %d",GetLastError());             // istek gönd
//--- işlemle ilgili veriler
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
  }
//+------------------------------------------------------------------+
```

Bekleyen Emri Değiştir

Bir bekleyen emrin fiyatlarını değiştirmek için alım-satım emri. Şu 7 alanın belirtilmesini gerektirir:

- action

- order
- price
- sl
- tp
- type_time
- expiration

Bekleyen emirlerin fiyat seviyelerinin değiştirilmesi için TRADE_ACTION_MODIFY işleminin bir örneği:

```
#define EXPERT_MAGIC 123456  // uzmanın tanıtıcı değeri (MagicNumber)
//+------------------------------------------------------------------+
//| Bekleyen emirlerin değiştirilmesi                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- alım-satım isteğininin ve istek sonucunun bildirimini yap ve başlat
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // bekleyen emirlerin toplam sayısı
//--- tüm bekleyen emirler için tekrarla
   for(int i=0; i<total; i++)
     {
      //--- emir parametreleri
      ulong  order_ticket=OrderGetTicket(i);                          // emir fişi
      string order_symbol=Symbol();                                   // sembol
      int    digits=(int)SymbolInfoInteger(order_symbol,SYMBOL_DIGITS); // basamak sa
      ulong  magic=OrderGetInteger(ORDER_MAGIC);                      // emri giren
      double volume=OrderGetDouble(ORDER_VOLUME_CURRENT);             // emrin mevc
      double sl=OrderGetDouble(ORDER_SL);                             // emrin mevc
      double tp=OrderGetDouble(ORDER_TP);                             // emrin mevc
      ENUM_ORDER_TYPE type=(ENUM_ORDER_TYPE)OrderGetInteger(ORDER_TYPE); // emir tipi
      int offset = 50;                                                // puan cinsi
      double price;                                                   // emrin teti
      double point=SymbolInfoDouble(order_symbol,SYMBOL_POINT);       // puan değer
      //--- emirle ilgili çıktı verileri
      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);
      //---tanıtıcı değer (MagicNumber) eşleşiyorsa Kar Al ve Zarar Durdur seviyeleri
      if(magic==EXPERT_MAGIC && sl==0 && tp==0)
        {
         request.action=TRADE_ACTION_MODIFY;                          // alım-satım
         request.order = OrderGetTicket(i);                           // emir fişi
         request.symbol   =Symbol();                                  // sembol
         request.deviation=5;                                         // izin veriler
        //--- fiyat seviyesinin, Zarar Durur ve Kar Al seviyelerinin emrin türüne göre
         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);              // normali
           }
         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);               // normal
           }
         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);                  // normal
           }
         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);                  // normal
           }
         //--- isteği gönder
         if(!OrderSend(request,result))
            PrintFormat("OrderSend hatası %d",GetLastError());  // istek gönderilemyo
         //--- işlemle ilgili veriler
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
         //--- istek ve sonuç değerlerini sıfırla
         ZeroMemory(request);
         ZeroMemory(result);
        }
     }
  }
//+------------------------------------------------------------------+
```

Bekleyen emri sil

Bekleyen bir emri silmek için alım-satım emri. Şu 2 alanın belirtilmesini gerektirir::

- action
- order

Bekleyen emirlerin silinmesi için bir TRADE_ACTION_REMOVE örneği:

```
#define EXPERT_MAGIC 123456  // uzmanın tanıtıcı değeri (MagicNumber)
```

```
//+------------------------------------------------------------------+
//| Bekleyen emirlerin silinmesi                                     |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- alım-satım isteğininin ve istek sonucunun bildirimini yap ve başlat
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   int total=OrdersTotal(); // bekleyen emirlerin toplam sayısı
//--- tüm bekleyen emirler için tekrarla
   for(int i=total-1; i>=0; i--)
     {
      ulong  order_ticket=OrderGetTicket(i);                   // emir fişi
      ulong  magic=OrderGetInteger(ORDER_MAGIC);               // emri giren uzmanın t
      //--- tanıtıcı değer (MagicNumber) eşleşiyorsa
      if(magic==EXPERT_MAGIC)
        {
         //--- istek ve sonuç değerlerini sıfırla
         ZeroMemory(request);
         ZeroMemory(result);
         //--- işlem parametrelerinin ayarlanması
         request.action=TRADE_ACTION_REMOVE;                   // alım-satım işleminir
         request.order = order_ticket;                         // emir fişi
         //--- isteği gönder
         if(!OrderSend(request,result))
            PrintFormat("OrderSend hatası %d",GetLastError());  // istek gönderilemyo
         //--- işlemle ilgili veriler
         PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,
        }
     }
  }
//+------------------------------------------------------------------+
```

Ayrıca bakınız

Yapılar ve Sınıflar, Alım-satım fonksiyonları, Emir Özellikleri

Alım-satım isteği kontrolü sonucunun yapısı (MqlTradeCheckResult)

Bir alım-satım sunucusuna, bir alım-satım işlemi için istek göndermeden önce kontrol edilmesi önerilir. Kontrol edilen istek ve MqlTradeCheckResult yapı değişkeninin parametre olarak geçirildiği OrderCheck() fonksiyonu ile kontrol gerçekleştirilir. Kontrol sonucu bu değişkene yazılacaktır.

```
struct MqlTradeCheckResult
  {
   uint          retcode;             // Cevap kodu
   double        balance;             // İşlemin gerçekleştirilmesinden sonraki bakiye
   double        equity;              // İşlemin gerçekleştirilmesinden sonraki varlık
   double        profit;              // Yüzer kar
   double        margin;              // Marjin (teminat) gereklilikleri
   double        margin_free;         // Serbest teminat
   double        margin_level;        // Marjin seviyesi
   string        comment;             // Cevap kodunun yorumu (hata tarifi)
  };
```

Alanların Açıklaması

Alan	Açıklama
retcode	Dönüş kodu
balance	Alım-satım işleminden sonra gerçekleşecek bakiye değeri
equity	Alım-satım işleminden sonra gerçekleşecek varlık değeri
profit	Alım-satım işleminden sonra gerçekleşecek yüzer karın değeri
margin	Alım-satım işlemi için istenen teminat
margin_free	Alım-satım işleminin gerçekleşmesinden sonra geriye kalacak olan serbest teminat
margin_level	Alım-satım işleminin gerçekleşmesinden sonra ayarlanacak teminat seviyesi
yorum	Cevap kodunun yorumu, hata tarifi

Ayrıca Bakınız

Alım-Satım İsteğinin Yapısı, Mevcut Fiyatlar için Yapı, OrderSend, OrderCheck

Bir Alım-Satım İsteği Sonucunun Yapısı (MqlTradeResult)

Alım-satım isteğinin bir sonucu olarak, a trade server returns data about the trade request processing result as a special predefined structure of MqlTradeResult type.

```
struct MqlTradeResult
  {
   uint     retcode;          // İşlem dönüş kodu
   ulong    deal;             // İşlem fişi, eğer gerçekleşmişse
   ulong    order;            // Emir fişi, eğer işleme konulmuşsa
   double   volume;           // Aracı kurum tarafından onaylanmış işlem hacmi
   double   price;            // Aracı kurum tarafından onaylanmış sözleşme fiyatı
   double   bid;              // Mevcut Satış fiyatı
   double   ask;              // Mevcut Alış fiyatı
   string   comment;          // Aracı kurum yorumu (varsayılan olarak işlem açıklamas
   int      request_id;       // İstek kimliği (tanımlayıcısı), terminal tarafından gö
  };
```

Alanların açıklaması

Alan	Açıklama
retcode	Bir alım-satım sunucusunun dönüş kodu
deal	İşlem fişi (bir işlem gerçekleştirilmişse). Bu, TRADE_ACTION_DEAL tipinde bir alım-satım işlemi için mevcut bulunmaktadır
order	Emir fişi, eğer bir fiş yerleştirilmişse. TRADE_ACTION_PENDING tipinde bir alım satım işlemi için mevcuttur
volume	Aracı kurum tarafından onaylanan işlem hacmi. Emir tipine bağlıdır
price	Aracı kurum tarafından onaylanan işlem fiyatı. Alım-satım isteğinin ve/veya alım-satım işleminin *deviation* alanına bağlıdır
bid	Mevcut piyasa satış fiyatı (yeniden fiyatlandırma değeri)
ask	Mevcut piyasa alış fiyatı (yeniden fiyatlandırma değeri)
yorum	Aracı kurumun işlem yorumu (ön tanımlı olarak işlem açıklaması ile doldurulur bir alım-satım sunucusunun dönüş kodu)
request_id	İstek kimliği, alım-satım sunucusuna gönderim yapılırken terminal tarafından ayarlanır.

Alım-satım işlemi, MqlTradeResult tipinde bir değişkene dönüş yapar; bu değişken, alım-satım işlemleri gerçekleştirmek için OrderSend() fonksiyonuna geçirilir.

Terminal, request_id alanındaki istek kimliğini, OrdersSend() ve OrderSendAsync() fonksiyonlarını kullanarak alım-satım sunucusuna gönderirken düzeltir. Terminal, gerçekleştirilmiş faaliyetler hakkında alım-satım sunucusundan mesajlar alır ve bunları OnTradeTransaction() fonksiyonu ile ibraz eder, fonksiyon şu bileşenleri parametre olarak içerir:

- MqlTradeTransaction yapısındaki alım-satım faaliyetinin tarifi;

- OrderSend() veya OrdersSendAsync() fonksiyonlarından gönderilen alım-satım isteğinin tarifi (açıklaması). İsteğin kendisi ve request_id terminal belleğinde saklanırken, istek kimliği terminal tarafından sunucuya gönderilir;
- Alım-satım işleminin gerçekleştirilmesi, bu isteğin kimliğini taşıyan request_id alanını içeren MqlTradeResult yapısıyla sonuçlanır.

OnTradeTransaction() fonksiyonu üç giriş parametresi alır ama son ikisi sadece TRADE_TRANSACTION_REQUEST tipi faaliyetler için analiz edilmelidir. Diğer tüm durumlarda, alım-satım isteğinin ve onun gerçekleşme sonucunun verileri doldurulmaz. Parametrelerin analiz örnekleri, Bir Alım-Satım İsteğinin Yapısı başlığı altında bulunabilir.

request_id alanının terminal tarafından alım-satım isteği için - istek sunucuya gönderirken - ayarlanması, temel olarak OrderSendAsync() asenkron fonksiyonu ile çalışmak için düşünülmüştür. Bu tanımlayıcı (kimlik), gerçekleşen eylem (OrderSend veya OrderSendAsync fonksiyonlarının çağrısı) ile bu eylemin OnTradeTransaction() fonksiyonuna gönderilen sonucunun ilişkilendirilmesini sağlar.

Örnek:

```
//+------------------------------------------------------------------+
//| Sonuç işlemeli bir alım-satım isteği gönder                      |
//+------------------------------------------------------------------+
bool MyOrderSend(MqlTradeRequest request,MqlTradeResult result)
  {
//--- son hatanın kodunu sıfırla
   ResetLastError();
//--- isteği gönder
   bool success=OrderSend(request,result);
//--- sonuç başarısızsa - neden olduğunu öğren
   if(!success)
     {
      int answer=result.retcode;
      Print("TradeLog: Alım-satım isteği alanı. Hata = ",GetLastError());
      switch(answer)
        {
         //--- yeniden fiyatlandırma
         case 10004:
           {
            Print("TRADE_RETCODE_REQUOTE");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
           }
         //--- emir, sunucu tarafından kabul edilmedi
         case 10006:
           {
            Print("TRADE_RETCODE_REJECT");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
           }
```

```
         //--- geçersiz fiyat
         case 10015:
           {
            Print("TRADE_RETCODE_INVALID_PRICE");
            Print("request.price = ",request.price,"   result.ask = ",
                  result.ask," result.bid = ",result.bid);
            break;
           }
         //--- geçersiz SL ve/veya TP
         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;
           }
         //--- geçersiz hacim
         case 10014:
           {
            Print("TRADE_RETCODE_INVALID_VOLUME");
            Print("request.volume = ",request.volume,"   result.volume = ",
                  result.volume);
            break;
           }
         //--- alım-satım işlemi için para yeterli değil
         case 10019:
           {
            Print("TRADE_RETCODE_NO_MONEY");
            Print("request.volume = ",request.volume,"   result.volume = ",
                  result.volume,"   result.comment = ",result.comment);
            break;
           }
         //--- başka bir sebep, sunucunun cevap kodunu çıktıla
         default:
           {
            Print("Diğer cevap = ",answer);
           }
        }
      //--- başarısız alım-satım isteği sonucu hakkında, false dönüşü yaparak uyar
      return(false);
     }
//--- OrderSend () 'true' dönüşü yapar - cevabı tekrarla
   return(true);
  }
```

Alım-Satım Faaliyeti Yapısı (MqlTradeTransaction)

Alım-satım hesabı üzerinde bazı kesin işlemler gerçekleştirildiği zaman hesabın durumu değişir. Bu eylemler şunları kapsamaktadır:

- OrderSend ve OrderSendAsync Fonksiyonları kullanılarak herhangi bir MQL5 uygulamasından yapılan alım-satım istekleri ve bunların ilerideki kullanımları;
- Terminalin grafiksel arayüzü ile yapılan alım satım istekleri ve bunların ilerideki kullanımları;
- Sunucu üzerindeki bekleyen emir ve durdurma emri aktivasyonu;
- İşlemlerin alım-satım sunucusu tarafından gerçekleştirilmesi.

Yukarıda sayılanlar eylemlerin sonucunda şu alım-satım faaliyetleri gerçekleşir:

- alım-satım isteğinin işlenmesi;
- açık emirlerin değişimi;
- emir geçmişinin değişimi;
- işlem geçmişinin değişimi;
- pozisyonların değişimi.

Örneğin, bir piyasa alım emri gönderirken, bu emir önce işlenir, hesap için uygun bir alım emri oluşturulur, sonra uygulanır ve açık emirler listesinden kaldırılır, ardından emir geçmişine eklenir, uygun bir işlem geçmişe eklenir ve yeni bir pozisyon oluşturulur. Tüm bu eylemler alım-satım faaliyetidir

Bir hesaba uygulanan alım-satım faaliyetlerini almak için, özel OnTradeTransaction() işleyicisi MQL5 içinde sağlanır. İşleyicinin ilk parametresi, alım-satım faaliyetlerini tanımlayan MqlTradeTransaction yapısını alır.

```
struct MqlTradeTransaction
  {
   ulong                     deal;             // İşlem fişi
   ulong                     order;            // Emir fişi
   string                    symbol;           // Alım-satım sembolünün ismi
   ENUM_TRADE_TRANSACTION_TYPE type;           // Alım-satım faaliyetinin tipi
   ENUM_ORDER_TYPE           order_type;       // Emir tipi
   ENUM_ORDER_STATE          order_state;      // Emir durumu
   ENUM_DEAL_TYPE            deal_type;        // İşlem tipi
   ENUM_ORDER_TYPE_TIME      time_type;        // Eylem periyoduyla Emir tipi
   datetime                  time_expiration;  // Emir zaman-aşımı zamanı
   double                    price;            // Fiyat
   double                    price_trigger;    // Dur-sınırı (stop limit) emri akt
   double                    price_sl;         // Stop Loss seviyesi
   double                    price_tp;         // Take Profit seviyesi
   double                    volume;           // Lot bazında hacim
   ulong                     position;         // Pozisyon fişi
   ulong                     position_by;      // Ters pozisyonun fişi
  };
```

Alanların Açıklaması

Alan	Açıklama
deal	İşlem fişi.
order	Emir fişi.
symbol	Faaliyetin gerçekleştirildiği alım-satım sembolünün ismi.
type	Alım-satım faaliyeti tipi. Değer, ENUM_TRADE_TRANSACTION_TYPE sayımının değerlerinden biri olabilir.
order_type	Alım-satım emrinin tipi. Değer, ENUM_ORDER_TYPE sayımının değerlerinden biri olabilir.
order_state	Alım-satım emrinin durumu. Değer, ENUM_ORDER_STATE sayımının değerlerinden biri olabilir.
deal_type	İşlem tipi. Değer, ENUM_DEAL_TYPE sayımının değerlerinden biri olabilir.
time_type	Zaman-aşımında emir tipi. Değer, ENUM_ORDER_TYPE_TIME sayımının değerlerinden biri olabilir.
time_expiration	Bekleyen emrin zaman-aşımı dönemi (ORDER_TIME_SPECIFIED ve ORDER_TIME_SPECIFIED_DAY tipleri için).
price	Fiyat. Bu, alım-satım faaliyetine bağlı olarak, bir emrin, bir işlemin veya bir pozisyonun fiyatı olabilir.
price_trigger	Dur-sınırı emri durdurma (aktivasyon) fiyatı (ORDER_TYPE_BUY_STOP_LIMIT ve ORDER_TYPE_SELL_STOP_LIMIT).
price_sl	Zarar Durdur fiyatı. Alım-satım faaliyetine bağlı olarak, bir emirle, bir işlemle veya bir pozisyon ile alakalı olabilir.
price_tp	Kar Al fiyatı. Alım-satım faaliyetine bağlı olarak, bir emirle, bir işlemle veya bir pozisyon ile alakalı olabilir.
volume	Lot bazında hacim. Alım-satım işlemine bağlı olarak, bir emrin, bir işlemin veya bir pozisyonun hacmini gösterebilir.
position	Faaliyetten etkilenen pozisyonun fişi.
position_by	Ters pozisyon fişi. Aynı sembol üzerindeki bir pozisyonu yine aynı sembol üzerindeki bir ters pozisyonla kapatmak için kullanılır.

Alım-satım faaliyetinin analizi için gereken temel parametre, faaliyetin **type** alanında belirtilen tipidir. Örneğin, eğer bir faaliyet TRADE_TRANSACTION_REQUEST tipinde ise (bir alım-satım isteğinin sunucu tarafından işlenmesinin sonucu alınmışsa) yapının sadece tek bir alanı doldurulacaktır - **type**. Diğer alanlar analiz edilmez. Bu durumda, **request** ve **result** parametrelerinin ek olarak OnTradeTransaction() işleyicisine geçirilmesini, aşağıda görüldüğü şekilde analiz edebiliriz.

Bir alım-satım işlemi ile ilgili verilere sahip olarak, alım-satım hesabındaki emirlerin, pozisyonların ve işlemlerin mevcut durumları hakkında yapılacak analize karar verebilirsiniz. Terminalden sunucuya gönderilen alım-satım isteği, birkaç yeni faaliyet oluşturabilir. Bunların terminale ulaşım öncelikleri garanti edilmez.

MqlTradeTransaction yapısı, (ENUM_TRADE_TRANSACTION_TYPE) alım-satım tipine bağlı olarak farklı yollarla doldurulur:

TRADE_TRANSACTION_ORDER_* ve TRADE_TRANSACTION_HISTORY_*

MqlTradeTransaction yapısının aşağıda belirtilen alanları, açık emirlerin işlenmesi (TRADE_TRANSACTION_ORDER_ADD, TRADE_TRANSACTION_ORDER_UPDATE ve TRADE_TRANSACTION_ORDER_DELETE) ve emir geçmişi (TRADE_TRANSACTION_HISTORY_ADD, TRADE_TRANSACTION_HISTORY_UPDATE, TRADE_TRANSACTION_HISTORY_DELETE) ile alakalı alım-satım faaliyetleri için doldurulur:

- order - emir fişi;
- symbol - emir sembolü ismi;
- type - alım-satım faaliyet tipi;
- order_type - emir tipi;
- orders_state - emrin mevcut durumu;
- time_type - emir zaman-aşımı tipi;
- time_expiration - emir zaman-aşımı zamanı (ORDER_TIME_SPECIFIED ve ORDER_TIME_SPECIFIED_DAY zaman-aşımı tiplerine sahip emirler için);
- price - müşteri tarafından belirlenen emir fiyatı;
- price_trigger - stop limit emri durdurma fiyatı (sadece ORDER_TYPE_BUY_STOP_LIMIT ve ORDER_TYPE_SELL_STOP_LIMIT tipleri için);
- price_sl - Stop Loss (kaybı durdur) emir fiyatı (emirde belirtilmişse doldurulur);
- price_tp - Take Profit (kar al) emir fiyatı (emirde belirtilmişse doldurulur);
- volume - mevcut emir hacmi (doldurulmaz). Başlangıç emir hacmi, HistoryOrders* fonksiyonu kullanılarak emir geçmişinde bulunabilir.
- position - işlem sonucunda değiştirilen veya kapatılan açık pozisyonun fişi. Sadece piyasa emirlerinde kullanılır, TRADE_TRANSACTION_ORDER_ADD için doldurulmaz.
- position_by - ters pozisyonun fişi. Sadece emir ile kapama için kullanılır (pozisyonu ters pozisyonla kapamak için).

TRADE_TRANSACTION_DEAL_*

MqlTradeTransaction yapısının aşağıda belirtilen alanları, işlemlerin işlenmesiyle alakalı alım-satım faaliyetleri (TRADE_TRANSACTION_DEAL_ADD, TRADE_TRANSACTION_DEAL_UPDATE and TRADE_TRANSACTION_DEAL_DELETE) için doldurulur:

- deal - işlem fişi;
- order - hangi işlemin gerçekleştiğine bağlı olarak, emir fişi;
- symbol - işlem sembolünün ismi;
- type - alım-satım faaliyet tipi;
- deal_type - işlem tipi;
- price - işlem fiyatı;
- price_sl - Zarar Durdur fiyatı (emirde belirtilmişse, hangi işlemin gerçekleştiğine bağlı olarak, doldurulur);
- price_tp - Kar Al fiyatı (emirde belirtilmişse, hangi işlemin gerçekleştiğine bağlı olarak, doldurulur);
- volume - lot bazında işlem hacmi.
- position - işlem sonucunda değiştirilen veya kapatılan açık pozisyonun fişi..
- position_by - ters pozisyonun fişi. Sadece emir ile kapama için kullanılır (pozisyonu ters pozisyonla kapamak için).

TRADE_TRANSACTION_POSITION

MqlTradeTransaction yapısının aşağıda belirtilen alanları, işlemlerin gerçekleştirilmesi ile ilgili olmayan; pozisyon değişimleri ile alakalı alım-satım faaliyetleri (TRADE_TRANSACTION_POSITION) için doldurulur:

- symbol - pozisyon sembol adı;
- type - alım-satım faaliyet tipi;
- deal_type - pzisyon tipi (DEAL_TYPE_BUY veya DEAL_TYPE_SELL);
- price - pozisyon açma fiyatı ağırlıklı ortalaması;
- price_sl - Zarar Durdur fiyatı;
- price_tp - Kar Al fiyatı;
- volume - eğer değiştirilmişse, lot bazında pozisyon fiyatı.

> Pozisyon değişimi (ekleme, değiştirme veya kapatma), bir işlemin gerçekleşmesinin sonucu olarak, TRADE_TRANSACTION_POSITION faaliyetinin ortaya çıkmasına neden olmaz.

TRADE_TRANSACTION_REQUEST

Bir alım-satım emrinin, sunucu tarafından işlendiği ve işlenme sonucunun alındığı (TRADE_TRANSACTION_REQUEST) gerçeğini açıklayan alım-satım işlemleri için, MqlTradeTransaction yapısının sadece bir alanı doldurulur:

- type - alım-satım faaliyet tipi;

> Bu gibi faaliyetler için sadece type alanı (alım-satım faaliyeti tipi) analiz edilmelidir. OnTradeTransaction fonksiyonunun ikinci ve üçüncü parametreleri(request ve result), ilave veriler için analiz edilmelidir.

Örnek:

```
input int MagicNumber=1234567;

//--- CTrade alım-satım sınıfını etkinleştir ve bu sınıfın değişkeninin bildirimini y
#include <Trade\Trade.mqh>
CTrade trade;
//--- bekleyen emrin yüklenmesi ve silinmesi için bayraklar
bool pending_done=false;
bool pending_deleted=false;
//--- bekleyen emir fişi burada saklanacak
ulong order_ticket;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- tüm emirlerimizi imlemesi (işaretlemesi) için MagicNumber ayarla
   trade.SetExpertMagicNumber(MagicNumber);
//--- alım-satım isteği, OrderSendAsync() fonksiyonu ile asenkron modda gönderilecek
   trade.SetAsyncMode(true);
//--- değişkeni sıfır ile başlat
   order_ticket=0;
```

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---bir bekleyen emrin yüklenmesi
   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);
      //--- BuyStop() fonksiyonu başarıyla gerçekleştirilmişse
      if(res)
        {
         pending_done=true;
         //--- Ctrade sınıfından, istek gönderiminin sonucunu al
         MqlTradeResult trade_result;
         trade.Result(trade_result);
         //--- gönderilen istek için request_id al
         uint request_id=trade_result.request_id;
         Print("İstek, bir bekleyen emir ayarlamak için gönderildi. Request_ID=",reque
         //--- emir fişi kaydediliyor (CTrade sınıfına gönderim yapılırken asenkron mo
         order_ticket=trade_result.order;
         //--- hepsi tamamlandı, OnTick() işleyicisinden erken çıkış
         return;
        }
     }
//--- bekleyen emri sil
   if(!pending_deleted)
      //--- ek kontrol
      if(pending_done && (order_ticket!=0))
        {
         //--- bekleyen emrin silinmesi deneniyor
         bool res=trade.OrderDelete(order_ticket);
         Print("OrderDelete=",res);
         //--- silme isteği başarıyla gönderildiğinde
         if(res)
           {
            pending_deleted=true;
            //--- istek yürütme sonucunu al
            MqlTradeResult trade_result;
            trade.Result(trade_result);
            //--- sonuçtan istek kimliğini (tanımlayıcısını) al
            uint request_id=trade_result.request_id;
            //--- günlükte görüntüle
            Print("İstek, bir bekleyen emrin silinmesi için gönderildi #",order_ticket
```

```
                  ". Request_ID=",request_id,
                  "\r\n");
            //--- istek sonucundan emir fişini düzelt
            order_ticket=trade_result.order;
           }
        }
//---
  }
//+------------------------------------------------------------------+
//| TradeTransaction function                                        |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//--- sayım değeri olarak faaliyet tipini al
   ENUM_TRADE_TRANSACTION_TYPE type=(ENUM_TRADE_TRANSACTION_TYPE)trans.type;
//--- eğer faaliyet, istek işleme sonucuysa, sadece ismi görüntülenir
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      Print(EnumToString(type));
      //--- işlenen isteğin dize ismini görüntüle
      Print("------------RequestDescription\r\n",RequestDescription(request));
      //--- istek sonucunun açıklamasını görüntüle
      Print("------------ResultDescription\r\n",TradeResultDescription(result));
      //--- bir sonraki OnTick() çağrısında silinmesi emir fişini kaydet
      if(result.order!=0)
        {
         //--- bir sonraki OnTick() çağrısında bu emri sil
         order_ticket=result.order;
         Print(" Bekleyen emir fişi ",order_ticket,"\r\n");
        }
     }
   else // başka tipteki faaliyetler için tam bir açıklama görüntüle
//--- alınan faaliyetin açıklamasını günlükte görüntüle
      Print("------------TransactionDescription\r\n",TransactionDescription(trans));

//---
  }
//+------------------------------------------------------------------+
//| Faaliyetin yazılı açıklamasına dönüş yapar                       |
//+------------------------------------------------------------------+
string TransactionDescription(const MqlTradeTransaction &trans)
  {
//---
   string desc=EnumToString(trans.type)+"\r\n";
   desc+="Symbol: "+trans.symbol+"\r\n";
   desc+="Deal ticket: "+(string)trans.deal+"\r\n";
   desc+="Deal type: "+EnumToString(trans.deal_type)+"\r\n";
```

```
   desc+="Order ticket: "+(string)trans.order+"\r\n";
   desc+="Order type: "+EnumToString(trans.order_type)+"\r\n";
   desc+="Order state: "+EnumToString(trans.order_state)+"\r\n";
   desc+="Order time type: "+EnumToString(trans.time_type)+"\r\n";
   desc+="Order expiration: "+TimeToString(trans.time_expiration)+"\r\n";
   desc+="Price: "+StringFormat("%G",trans.price)+"\r\n";
   desc+="Price trigger: "+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";
   desc+="Position: "+(string)trans.position+"\r\n";
   desc+="Position by: "+(string)trans.position_by+"\r\n";
//--- elde edilen dizeye dönüş yap
   return desc;
  }
//+------------------------------------------------------------------+
//| Alım-satım isteğinin yazılı açıklamasına dönüş yapar             |
//+------------------------------------------------------------------+
string RequestDescription(const MqlTradeRequest &request)
  {
//---
   string desc=EnumToString(request.action)+"\r\n";
   desc+="Symbol: "+request.symbol+"\r\n";
   desc+="Magic Number: "+StringFormat("%d",request.magic)+"\r\n";
   desc+="Order ticket: "+(string)request.order+"\r\n";
   desc+="Order type: "+EnumToString(request.type)+"\r\n";
   desc+="Order filling: "+EnumToString(request.type_filling)+"\r\n";
   desc+="Order time type: "+EnumToString(request.type_time)+"\r\n";
   desc+="Order expiration: "+TimeToString(request.expiration)+"\r\n";
   desc+="Price: "+StringFormat("%G",request.price)+"\r\n";
   desc+="Deviation points: "+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+="Comment: "+request.comment+"\r\n";
//--- elde edilen dizeye dönüş yap
   return desc;
  }
//+------------------------------------------------------------------+
//| İstek sonucunun yazılı açıklamasına dönüş yapar                  |
//+------------------------------------------------------------------+
string TradeResultDescription(const MqlTradeResult &result)
  {
//---
   string desc="Retcode "+(string)result.retcode+"\r\n";
   desc+="Request ID: "+StringFormat("%d",result.request_id)+"\r\n";
   desc+="Order ticket: "+(string)result.order+"\r\n";
   desc+="Deal ticket: "+(string)result.deal+"\r\n";
```

```
   desc+="Volume: "+StringFormat("%G",result.volume)+"\r\n";
   desc+="Price: "+StringFormat("%G",result.price)+"\r\n";
   desc+="Ask: "+StringFormat("%G",result.ask)+"\r\n";
   desc+="Bid: "+StringFormat("%G",result.bid)+"\r\n";
   desc+="Comment: "+result.comment+"\r\n";
//--- elde edilen dizeye dönüş yap
   return desc;
  }
```

Ayrıca bakınız

Alım-Satım Faaliyet Tipleri, OnTradeTransaction()

Mevcut fiyatları almak için bir yapı (MqlTick)

Bu yapı, ilgili sembolün son fiyatlarını içerir. Mevcut fiyatlar hakkındaki en çok istenen bilgilerin hızlı şekilde elde edilmesi için tasarlanmıştır.

```
struct MqlTick
  {
   datetime     time;          // Son fiyat güncellemesinin zamanı
   double       bid;           // Mevcut Satış fiyatı
   double       ask;           // Mevcut Alış fıyatı
   double       last;          // Son işlem fiyatı (Last)
   ulong        volume;        // Son fiyat için hacim
   long         time_msc;      // Son fiyatın milisaniye cinsinden güncellenme zamanı
   uint         flags;         // Tik bayrakları
   double       volume_real;   // Mevcut Son fiyat için daha yüksek doğruluklu hacim
  };
```

MqlTick tipi değişken SymbolInfoTick() fonksiyonunun bir çağrısı ile, Alış, Satış, Son fiyat ve Hacim değerlerinin alınmasını sağlar.

Her bir tik parametresi, önceki tike göre değişim olup olmadığına bakılmaksızın doldurulur. Böylece tik geçmişindeki daha önceki verilere bakmaksızın geçmişteki belli bir ana dair doğru fiyat verilerine ulaşmak mümkündür. Örneğin, yeni tik üzerinde sadece Satış verisi değişmiş olsa da, Alış, Hacim, vb. diğer veriler de yapı içerisinde yer alacaktır.

Hangi verinin değiştiğini görmek için tik bayraklarını kullanabilirsiniz:

- TICK_FLAG_BID - Satış fiyatı verisi değişti
- TICK_FLAG_ASK - Alış fiyatı verisi değişti
- TICK_FLAG_LAST - son fiyat verisi değişti
- TICK_FLAG_VOLUME - hacim değişti
- TICK_FLAG_BUY - alış işlemi sonucunda oluşan tik verisi
- TICK_FLAG_SELL - satış işlemi sonucunda oluşan tik verisi

Örnek:

```
void OnTick()
  {
   MqlTick last_tick;
//---
   if(SymbolInfoTick(Symbol(),last_tick))
     {
      Print(last_tick.time,": Satış = ",last_tick.bid,
            " Alış = ",last_tick.ask,"  Hacim = ",last_tick.volume);
     }
   else Print("SymbolInfoTick() başarısız oldu, hata = ",GetLastError());
//---
  }
```

Ayrıca Bakınız

Yapılar ve Sınıflar, CopyTicks(), SymbolInfoTick()

Ekonomik Takvim yapıları

Bu bölümde, doğrudan MetaTrader platformunda bulunan ekonomik takvim ile çalışma adına yapıları açıklanmaktadır. Ekonomik takvim, makroekonomik göstergelerin açıklamalarını, yayımlama tarihlerini ve önem derecelerini içeren hazır bir ansiklopedidir. Makroekonomik göstergelerin ilgili değerleri, yayımlandığı anda MetaTrader platformuna gönderilir ve bu değerler, gerekli göstergeleri ülkeler, para birimleri ve önem açısından görsel olarak izlemenizi sağlayan etiketler halinde görüntülenir.

Ekonomik takvim fonksiyonları, gelen olayların otomatik olarak gerekli ülke/döviz pariteleri perspektifinden özel önem kriterlerine göre analiz edilmesini sağlar.

Ülke açıklamaları MqlCalendarCountry yapısı tarafından ayarlanır. Bu yapı, CalendarCountryById() ve CalendarCountries() fonksiyonlarında kullanılır.

```
struct MqlCalendarCountry
  {
   ulong                             id;                // ülke ID'si (ISO 3166-
   string                            name;              // yazısal olarak ülke a
   string                            code;              // kod olarak ülke adı
   string                            currency;          // ülke para birimi kodu
   string                            currency_symbol;   // ülke para birimi semb
   string                            url_name;          // mql5.com web sitesi U
  };
```

Olay açıklamaları MqlCalendarEvent yapısı tarafından ayarlanır. Bu yapı, CalendarEventById(), CalendarEventByCountry() ve CalendarEventByCurrency() fonksiyonlarında kullanılır.

```
struct MqlCalendarEvent
  {
   ulong                             id;                // olay ID'si
   ENUM_CALENDAR_EVENT_TYPE          type;              // ENUM_CALENDAR_EVENT_T
   ENUM_CALENDAR_EVENT_SECTOR        sector;            // olayın ilgili olduğu
   ENUM_CALENDAR_EVENT_FREQUENCY     frequency;         // olay sıklığı
   ENUM_CALENDAR_EVENT_TIMEMODE      time_mode;         // olayın zamanı modu
   ulong                             country_id;        // ülke ID'si
   ENUM_CALENDAR_EVENT_UNIT          unit;              // ekonomik gösterge değ
   ENUM_CALENDAR_EVENT_IMPORTANCE    importance;        // olayın önemi
   ENUM_CALENDAR_EVENT_MULTIPLIER    multiplier;        // ekonomik gösterge değ
   uint                              digits;            // ondalık basamak sayıs
   string                            source_url;        // olayın yayımlandığı I
   string                            event_code;        // olay kodu
   string                            name;              // olayın terminal dilin
  };
```

Olay değerleri MqlCalendarValue yapısı tarafından ayarlanır. Bu yapı, CalendarValueById(), CalendarValueHistoryByEvent(), CalendarValueHistory(), CalendarValueLastByEvent() ve CalendarValueLast() fonksiyonlarında kullanılır.

```
struct MqlCalendarValue
  {
   ulong                             id;                // değerin kimliği
   ulong                             event_id;          // olayın kimliği
   datetime                          time;              // olayın zamanı ve tar
   datetime                          period;            // olayın raporlama dön
   int                               revision;          // yayımlanan göstergen
   long                              actual_value;      // ppm cinsinden açıkla
   long                              prev_value;        // ppm cinsinden önceki
   long                              revised_prev_value; // ppm cinsinden revize
   long                              forecast_value;    // ppm cinsinden beklent
   ENUM_CALENDAR_EVENT_IMPACT        impact_type;       // döviz kuru üzerindek
  //--- değerleri kontrol eden fonksiyonlar
   bool                         HasActualValue(void) const;   // actual_value belirle
   bool                         HasPreviousValue(void) const; // prev_value belirlenm
   bool                         HasRevisedValue(void) const;  // revised_prev_value b
   bool                         HasForecastValue(void) const; // forecast_value belir
  //--- değerleri alan fonksiyonlar
   double                       GetActualValue(void) const;   // actual_value değerin
   double                       GetPreviousValue(void) const; // prev_value değerini
   double                       GetRevisedValue(void) const;  // revised_prev_value d
   double                       GetForecastValue(void) const; // forecast_value değer
  };
```

MqlCalendarValue yapısı, *actual_value*, *forecast_value*, *prev_value* ve *revised_prev_value* alanlarındaki değerleri kontrol etmek ve ayarlamak için yöntemler sağlar. Değer belirlenmezse, alan LONG_MIN (-9223372036854775808) olarak ayarlanır.

Lütfen bu alanlarda bulunan değerlerin bir milyon ile çarpıldığını unutmayın. Bunun anlamı, CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent ve CalendarValueLast fonksiyonlarını kullanarak MqlCalendarValue'da değerler aldığınızda, alandaki değerlerin LONG_MIN'e eşit olup olmadığını kontrol etmeniz; alanda bir değer belirlenmişse, değeri elde etmek için değeri 1.000.000'a bölmeniz gerektiği anlamına gelir. Değerleri elde etmenin diğer bir yöntemi de MqlCalendarValue yapısının fonksiyonlarını kullanarak değerleri kontrol etmek ve elde etmektir.

Takvim olaylarını yönetmeye bir örnek:

```
//--- Takvim olaylarını tam sayılar yerine gerçek değerlerle elde etmek için bir yapı
struct AdjustedCalendarValue
  {
   ulong                             id;                // değerin kimliği
   ulong                             event_id;          // olayın kimliği
   datetime                          time;              // olayın zamanı ve tar
   datetime                          period;            // olayın raporlama dön
   int                               revision;          // yayımlanan göstergen
```

```
   double                              actual_value;          // açıklanan değer
   double                              prev_value;            // önceki değer
   double                              revised_prev_value;    // revize edilmiş önceki
   double                              forecast_value;        // beklenti değeri
   ENUM_CALENDAR_EVENT_IMPACT          impact_type;           // döviz kuru üzerindeki
  };
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
//--- AB için ülke kodu (ISO 3166-1 Alpha-2)
   string EU_code="EU";
//--- tüm AB olaylarının değerlerini al
   MqlCalendarValue values[];
//--- olayların alındığı aralığın sınırlarını belirle
   datetime date_from=D'01.01.2021';  // 2021'deki tüm olayları al
   datetime date_to=0;                // 0, henüz gerçekleşmemiş olanlar dahil tüm bil
//--- 2021 yılından bu yana AB olay geçmişini talep et
   if(!CalendarValueHistory(values, date_from, date_to, EU_code))
     {
      PrintFormat("Hata! country_code=%s için olaylar alınamadı", EU_code);
      PrintFormat("Hata kodu: %d", GetLastError());
      return;
     }
   else
      PrintFormat("country_code=%s için olayların değerleri alındı: %d",
                  EU_code, ArraySize(values));
//--- Günlük çıktısı için dizinin boyutunu küçült
   if(ArraySize(values)>5)
      ArrayResize(values, 5);
//--- olayların değerlerini, kontrol etmeden veya gerçek değerlere dönüştürmeden, oldu
   Print("Takvim değerlerini oldukları gibi elde et");
   ArrayPrint(values);

//--- alan değerlerini kontrol et ve gerçek değerlere dönüştür
//--- değerleri kontrol etmek ve almak için 1. seçenek
   AdjustedCalendarValue values_adjusted_1[];
   int total=ArraySize(values);
   ArrayResize(values_adjusted_1, total);
//--- değerleri kontroller ve ayarlamalar ile kopyala
   for(int i=0; i<total; i++)
     {
      values_adjusted_1[i].id=values[i].id;
      values_adjusted_1[i].event_id=values[i].event_id;
      values_adjusted_1[i].time=values[i].time;
      values_adjusted_1[i].period=values[i].period;
      values_adjusted_1[i].revision=values[i].revision;
```

```
      values_adjusted_1[i].impact_type=values[i].impact_type;
      //--- değerleri kontrol et ve 1.000.000'a böl
      if(values[i].actual_value==LONG_MIN)
         values_adjusted_1[i].actual_value=double("nan");
      else
         values_adjusted_1[i].actual_value=values[i].actual_value/1000000.;

      if(values[i].prev_value==LONG_MIN)
         values_adjusted_1[i].prev_value=double("nan");
      else
         values_adjusted_1[i].prev_value=values[i].prev_value/1000000.;

      if(values[i].revised_prev_value==LONG_MIN)
         values_adjusted_1[i].revised_prev_value=double("nan");
      else
         values_adjusted_1[i].revised_prev_value=values[i].revised_prev_value/1000000.

      if(values[i].forecast_value==LONG_MIN)
         values_adjusted_1[i].forecast_value=double("nan");
      else
         values_adjusted_1[i].forecast_value=values[i].forecast_value/1000000.;
     }
   Print("Takvim değerlerini kontrol etmenin ve almanın ilk yöntemi");
   ArrayPrint(values_adjusted_1);

//--- değerleri kontrol etmek ve almak için 2. seçenek
   AdjustedCalendarValue values_adjusted_2[];
   ArrayResize(values_adjusted_2, total);
//--- değerleri kontroller ve ayarlamalar ile kopyala
   for(int i=0; i<total; i++)
     {
      values_adjusted_2[i].id=values[i].id;
      values_adjusted_2[i].event_id=values[i].event_id;
      values_adjusted_2[i].time=values[i].time;
      values_adjusted_2[i].period=values[i].period;
      values_adjusted_2[i].revision=values[i].revision;
      values_adjusted_2[i].impact_type=values[i].impact_type;
      //--- değerleri kontrol et ve al
      if(values[i].HasActualValue())
         values_adjusted_2[i].actual_value=values[i].GetActualValue();
      else
         values_adjusted_2[i].actual_value=double("nan");

      if(values[i].HasPreviousValue())
         values_adjusted_2[i].prev_value=values[i].GetPreviousValue();
      else
         values_adjusted_2[i].prev_value=double("nan");

      if(values[i].HasRevisedValue())
```

```
         values_adjusted_2[i].revised_prev_value=values[i].GetRevisedValue();
      else
         values_adjusted_2[i].revised_prev_value=double("nan");

      if(values[i].HasForecastValue())
         values_adjusted_2[i].forecast_value=values[i].GetForecastValue();
      else
         values_adjusted_2[i].forecast_value=double("nan");
     }
   Print("Takvim değerlerini kontrol etmenin ve almanın ikinci yöntemi");
   ArrayPrint(values_adjusted_2);

//--- değerleri almak için 3. seçenek - kontroller olmadan
   AdjustedCalendarValue values_adjusted_3[];
   ArrayResize(values_adjusted_3, total);
//--- değerleri kontroller ve ayarlamalar ile kopyala
   for(int i=0; i<total; i++)
     {
      values_adjusted_3[i].id=values[i].id;
      values_adjusted_3[i].event_id=values[i].event_id;
      values_adjusted_3[i].time=values[i].time;
      values_adjusted_3[i].period=values[i].period;
      values_adjusted_3[i].revision=values[i].revision;
      values_adjusted_3[i].impact_type=values[i].impact_type;
      //--- kontroller olmadan değerleri al
      values_adjusted_3[i].actual_value=values[i].GetActualValue();
      values_adjusted_3[i].prev_value=values[i].GetPreviousValue();
      values_adjusted_3[i].revised_prev_value=values[i].GetRevisedValue();
      values_adjusted_3[i].forecast_value=values[i].GetForecastValue();
     }
   Print("Takvim değerlerini almanın üçüncü yöntemi - kontroller olmadan");
   ArrayPrint(values_adjusted_3);
  }
/*
   country_code=EU için olayların değerlerini aldık: 1051
  Takvim değerlerini oldukları gibi elde et
         [id] [event_id]              [time]            [period] [revision]        [act
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0 -922337203
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0
  Takvim değerlerini kontrol etmenin ve almanın ilk yöntemi
         [id] [event_id]              [time]            [period] [revision] [actual_va
   [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3       55.2
   [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0      143.1
   [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
   [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0       11.0
   [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0        3.1
```

```
  Takvim değerlerini kontrol etmenin ve almanın ikinci yöntemi
        [id] [event_id]              [time]            [period] [revision] [actual_va
  [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3       55.2
  [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0      143.1
  [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
  [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0       11.0
  [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0        3.1
  Takvim değerlerini almanın üçüncü yöntemi - kontroller olmadan
        [id] [event_id]              [time]            [period] [revision] [actual_va
  [0] 144520  999500001 2021.01.04 12:00:00 2020.12.01 00:00:00          3       55.2
  [1] 144338  999520001 2021.01.04 23:30:00 2020.12.29 00:00:00          0      143.1
  [2] 147462  999010020 2021.01.04 23:45:00 1970.01.01 00:00:00          0
  [3] 111618  999010018 2021.01.05 12:00:00 2020.11.01 00:00:00          0       11.0
  [4] 111619  999010019 2021.01.05 12:00:00 2020.11.01 00:00:00          0        3.1
*/
```

Olay sıklığı, MqlCalendarEvent yapısında belirtilir. Olası değerler şu listede belirtilmiştir: **ENUM_CALENDAR_EVENT_FREQUENCY**

ID	Açıklama
CALENDAR_FREQUENCY_NONE	Yayımlama sıklığı ayarlanmadı
CALENDAR_FREQUENCY_WEEK	Haftada bir yayın
CALENDAR_FREQUENCY_MONTH	Ayda bir yayın
CALENDAR_FREQUENCY_QUARTER	3 ayda(çeyrekte) bir yayın
CALENDAR_FREQUENCY_YEAR	Yılda bir yayın
CALENDAR_FREQUENCY_DAY	Günde bir yayın

Olay tipi, MqlCalendarEvent yapısında belirtilir. Olası değerler şu listede belirtilmiştir: **ENUM_CALENDAR_EVENT_TYPE**

ID	Açıklama
CALENDAR_TYPE_EVENT	Olay (toplantı, konuşma vb.)
CALENDAR_TYPE_INDICATOR	Gösterge
CALENDAR_TYPE_HOLIDAY	Tatil

Bir olayın ilişkili olduğu ekonominin sektörü, MqlCalendarEvent yapısında belirtilir. Olası değerler şu listede belirtilmiştir: **ENUM_CALENDAR_EVENT_SECTOR**

ID	Açıklama
CALENDAR_SECTOR_NONE	Sektör ayarlanmadı

ID	Açıklama
CALENDAR_SECTOR_MARKET	Piyasa, borsa
CALENDAR_SECTOR_GDP	Gayri Safi Yurtiçi Hasıla (GSYİH)
CALENDAR_SECTOR_JOBS	İş piyasası
CALENDAR_SECTOR_PRICES	Fiyatlar
CALENDAR_SECTOR_MONEY	Para
CALENDAR_SECTOR_TRADE	Alım-satım
CALENDAR_SECTOR_GOVERNMENT	Devlet
CALENDAR_SECTOR_BUSINESS	Ticari faaliyet (iş)
CALENDAR_SECTOR_CONSUMER	Tüketim
CALENDAR_SECTOR_HOUSING	Konut
CALENDAR_SECTOR_TAXES	Vergiler
CALENDAR_SECTOR_HOLIDAYS	Tatiller

Olayın önemi, MqlCalendarEvent yapısında belirtilir. Olası değerler şu listede belirtilmiştir: **ENUM_CALENDAR_EVENT_IMPORTANCE**

ID	Açıklama
CALENDAR_IMPORTANCE_NONE	Önem ayarlanmadı
CALENDAR_IMPORTANCE_LOW	Düşük önem
CALENDAR_IMPORTANCE_MODERATE	Orta önem
CALENDAR_IMPORTANCE_HIGH	Yüksek önem

Olay değerlerinin görüntülenmesinde kullanılan ölçü birimi tipi, MqlCalendarEvent yapısında belirtilir. Olası değerler şu listede belirtilmiştir: **ENUM_CALENDAR_EVENT_UNIT**

ID	Açıklama
CALENDAR_UNIT_NONE	Ölçüm birimi ayarlanmadı
CALENDAR_UNIT_PERCENT	Yüzde
CALENDAR_UNIT_CURRENCY	Ulusal para birimi
CALENDAR_UNIT_HOUR	Saatler
CALENDAR_UNIT_JOB	İşler
CALENDAR_UNIT_RIG	Sondaj kuleleri

ID	Açıklama
CALENDAR_UNIT_USD	USD
CALENDAR_UNIT_PEOPLE	İnsanlar
CALENDAR_UNIT_MORTGAGE	İpotek kredileri
CALENDAR_UNIT_VOTE	Oylar
CALENDAR_UNIT_BARREL	Variller
CALENDAR_UNIT_CUBICFEET	Kübik kadem
CALENDAR_UNIT_POSITION	Ticari Olmayan Net Pozisyonlar
CALENDAR_UNIT_BUILDING	Binalar

Bazı durumlarda, ekonomik parametre değerleri MqlCalendarEvent yapısında ayarlanmış bir çarpan gerektirir. Olası çarpanlar şu listede belirtilmiştir: **ENUM_CALENDAR_EVENT_MULTIPLIER**

ID	Açıklama
CALENDAR_MULTIPLIER_NONE	Çarpan ayarlanmadı
CALENDAR_MULTIPLIER_THOUSANDS	Binler
CALENDAR_MULTIPLIER_MILLIONS	Milyonlar
CALENDAR_MULTIPLIER_BILLIONS	Milyarlar
CALENDAR_MULTIPLIER_TRILLIONS	Trilyonlar

Olayın ulusal para birimi oranı üzerindeki potansiyel etkisi, MqlCalendarValue yapısında gösterilir. Olası değerler şu listede belirtilmiştir: **ENUM_CALENDAR_EVENT_IMPACT**

ID	Açıklama
CALENDAR_IMPACT_NA	Etki ayarlanmadı
CALENDAR_IMPACT_POSITIVE	Pozitif etki
CALENDAR_IMPACT_NEGATIVE	Negatif etki

Olayın zamanı, MqlCalendarEvent yapısında belirtilir. Olası değerler şu listede belirtilmiştir: **ENUM_CALENDAR_EVENT_TIMEMODE**

ID	Açıklama
CALENDAR_TIMEMODE_DATETIME	Kaynak, bir olayın kesin zamanını yayımlamaktadır
CALENDAR_TIMEMODE_DATE	Olay tüm gün sürmektedir

ID	Açıklama
CALENDAR_TIMEMODE_NOTIME	Kaynak, olay için hiçbir zaman yayımlamamaktadır
CALENDAR_TIMEMODE_TENTATIVE	Kaynak, olayın kesin zamanı yerine bir olay günü yayımlamaktadır. Olayın gerçekleşmesi akabinde zaman belirlenir.

Ayrıca bakınız

Ekonomik Takvim

Hata ve Uyarı Kodları

Bu bölüm şu tarifleri içermektedir:

- Alım-satım sunucusunun dönüş kodları - OrderSend() fonksiyonu ile gönderilen alım-satım isteğinin analiz sonuçları;
- Derleyici Uyarıları - derleme sırasında görünen uyarı mesajlarının kodları (hata değil);
- Derleme hataları - başarısız bir derleme girişimindeki hata mesajlarının kodları;
- Çalışma zamanı hataları - GetLastError() fonksiyonu kullanılarak elde edilen, MQL5 programlarının performanslarındaki hata kodları.

Alım-Satım Sunucusunun Dönüş Kodları

Alım-satım işlemlerinin için yapılan tüm istekler OrderSend() fonksiyonu kullanılarak MqlTradeRequest yapısı şeklinde gönderilir. Fonksiyonun uygulama sonucu MqlTradeResult yapısına yerleştirilir. Bu yapı *retcode* alanında sunucunun dönüş kodunu içerir.

Kod	Sabit	Açıklama
10004	TRADE_RETCODE_REQUOTE	Yeniden fiyatlandırma (requote)
10006	TRADE_RETCODE_REJECT	İstek reddedildi
10007	TRADE_RETCODE_CANCEL	İstek kullanıcı tarafından iptal edildi
10008	TRADE_RETCODE_PLACED	Emir işlendi
10009	TRADE_RETCODE_DONE	İstek tamamlandı
10010	TRADE_RETCODE_DONE_PARTIAL	İsteğin sadece bir kısmı tamamlandı
10011	TRADE_RETCODE_ERROR	İstek işleme hatası
10012	TRADE_RETCODE_TIMEOUT	İstek zaman-aşımı nedeniyle iptal edildi
10013	TRADE_RETCODE_INVALID	Geçersiz istek
10014	TRADE_RETCODE_INVALID_VOLUME	Geçersiz istek hacmi
10015	TRADE_RETCODE_INVALID_PRICE	Geçersiz istek fiyatı
10016	TRADE_RETCODE_INVALID_STOPS	İstek içinde geçersiz stop (durdurma) seviyeleri
10017	TRADE_RETCODE_TRADE_DISABLED	Alım-satım işlemleri devre dışı
10018	TRADE_RETCODE_MARKET_CLOSED	Piyasa kapalı
10019	TRADE_RETCODE_NO_MONEY	İsteği tamamlayabilecek yeterli para yok
10020	TRADE_RETCODE_PRICE_CHANGED	Fiyatlar değişti
10021	TRADE_RETCODE_PRICE_OFF	İstek içinde işlenebilecek bir fiyat teklifi yok
10022	TRADE_RETCODE_INVALID_EXPIRATION	İstek içinde geçersiz zaman-aşımı tarihi
10023	TRADE_RETCODE_ORDER_CHANGED	Emir durumu değişti
10024	TRADE_RETCODE_TOO_MANY_REQUESTS	İsteklerin sıklığı fazla
10025	TRADE_RETCODE_NO_CHANGES	İstekte değişim yok
10026	TRADE_RETCODE_SERVER_DISABLES_AT	Otomatik alım-satım, sunucu tarafından devre dışı bırakıldı
10027	TRADE_RETCODE_CLIENT_DISABLES_AT	Otomatik alım-satım, müşteri terminali tarafından devre dışı bırakıldı
10028	TRADE_RETCODE_LOCKED	İstek işlem için kilitlendi
10029	TRADE_RETCODE_FROZEN	Emir veya pozisyon dondurulmuş

Kod	Sabit	Açıklama
10030	TRADE_RETCODE_INVALID_FILL	Geçersiz emir işleme tipi
10031	TRADE_RETCODE_CONNECTION	Alım-satım sunucusuyla bağlantı yok
10032	TRADE_RETCODE_ONLY_REAL	Sadece gerçek hesaplarda işleme izin veriliyor
10033	TRADE_RETCODE_LIMIT_ORDERS	Bekleyen emirlerin sayısı sınıra ulaştı
10034	TRADE_RETCODE_LIMIT_VOLUME	Sembol üzerindeki emirler ve pozisyonlar için hacim sınırına ulaşıldı
10035	TRADE_RETCODE_INVALID_ORDER	Yanlış veya geçersiz emir tipi
10036	TRADE_RETCODE_POSITION_CLOSED	Belirtilmiş POSITION_IDENTIFIER özelliğine sahip olan pozisyon kapatılmış
10038	TRADE_RETCODE_INVALID_CLOSE_VOLUME	Kapanış hacmi mevcut pozisyon hacmini geçiyor
10039	TRADE_RETCODE_CLOSE_ORDER_EXIST	Belirtilen pozisyon için zaten bir kapanış emri var. Bu durum hedge'li sistemde ortaya çıkabilir: • Bir pozisyonun ters yönlü başka bir pozisyonla kapatılması denendiğinde, önceden tanımlanan bir kapanış emri mevcutsa • Pozisyonun tamamen veya kısmen kapatılması denendiğinde, önceden tanımlanan ve yeni yerleştirilen kapanış emirlerinin hacimlerinin mevcut pozisyon hacmini aşması durumunda
10040	TRADE_RETCODE_LIMIT_POSITIONS	Hesap üzerinde aynı anda açık olabilecek posizyonların sayısı sunucu ayarlarında sınırlandırılmış olabilir. Bu sınıra ulaşıldığında her yeni emir girişimi için sunucu tarafından TRADE_RETCODE_LIMIT_POSITIONS şeklinde bir hata dönüşü yapılır. Sınırlama, pozisyon muhasebesi sistemine bağlı olarak değişim gösterir: • Netleştirme — açık pozisyonların sayısı göz önüne alınır. Sınıra ulaşıldığında, pozisyon sayısını artıracak yeni emirler platform tarafından devre-dışı bırakılır. Aslında, platform yeni emirlere sadece üzerinde zaten açık pozisyon bulunan semboller için izin verir. Mevcut bekleyen emirler, gerçekleşmeleri durumunda pozisyon sayısını artırmayabilecekleri için gözardı edilir ve pozisyon sayısını artıramazlar

Kod	Sabit	Açıklama
		• Hedge — her bekleyen emrin tetiklenmesi yeni bir pozisyonla sonuçlanacağı için, bekleyen emirler de açık pozisyonlarla birlikte değerlendirilir. Sınıra ulaşıldığında, platform bekleyen emirleri de piyasa emirleri gibi devre dışı bırakır.
10041	TRADE_RETCODE_REJECT_CANCEL	Bekleyen emir isteği reddedildi, emir iptal edildi
10042	TRADE_RETCODE_LONG_ONLY	İstek reddedildi, çünkü sembol için"**sadece uzun pozisyon açılabilir**" kuralı ayarlanmış (POSITION_TYPE_BUY)
10043	TRADE_RETCODE_SHORT_ONLY	İstek reddedildi, çünkü sembol için"**sadece kısa pozisyon açılabilir**" kuralı ayarlanmış (POSITION_TYPE_SELL)
10044	TRADE_RETCODE_CLOSE_ONLY	İstek reddedildi, çünkü sembol için"**sadece pozisyon kapatılabilir**" kuralı ayarlanmış
10045	TRADE_RETCODE_FIFO_CLOSE	İstek reddedildi, çünkü "**Pozisyon kapatmaya yalnızca FIFO kuralına göre izin verilmektedir**" bayrağı alım-satım hesabı için ayarlanmıştır (ACCOUNT_FIFO_CLOSE=true)
10046	TRADE_RETCODE_HEDGE_PROHIBITED	İşlem hesabı için "**Sembol üzerinde zıt pozisyon açma devre dışıdır**" kuralı ayarlandığından, istek reddedildi. Örneğin, işlem hesabında Alış pozisyonu varsa, kullanıcı Satış pozisyonu açamaz veya bekleyen satış emri giremez. Kural yalnızca hedge muhasebe sistemine sahip hesaplara uygulanır (ACCOUNT_MARGIN_MODE=ACCOUNT_MARGIN_MODE_RETAIL_HEDGING).

Derleyici Uyarıları

Derleyici uyarıları sadece bilgi amaçlı gösterilirler ve hata mesajı niteliği taşımazlar.

Kod	Açıklama
21	datetime dizgisi içinde, eksik tarih kaydı
22	datetime dizgisi içinde, tarih için verilen sayı yanlış. İstenenler: Yıl 1970 <= X <= 3000 Ay 0 <X <= 12 Gün 0 <X <= 31/30/28 (29)....
23	datetime dizgisi içinde, zaman için verilen için sayı yanlış. İstenenler: Saat 0 <= X <24 Dakika 0 <= X <60
24	RGB biçiminde bilinmeyen renk: RGB bileşenlerinden biri 0'dan küçük veya 255'ten büyük
25	Kaçış dizisinin bilinmeyen bir karakteri. Bilinen: \n \r \t \\ \" \' \X \x
26	fonksiyon içindeki yerel değişkenlerin boyutu çok büyük (> 512Kb), sayıyı azaltın
29	Sayım daha önce tanımlanmış (tekrar) - elemanlar ilk tanıma eklenecek
30	Makronun yeniden tanımlanması
31	Değişken bildirilmiş ama hiçbir yerde kullanılmamış
32	Yapıcı, void tipinde olmalı
33	Yıkıcı, void tipinde olmalı
34	Sabit, tamsayı kapsamına uymuyor (X> _UI64_MAX | | X <_I64_MIN) ve double tipine dönüştürülecek
35	HEX çok uzun- 16 anlamlı karakterden fazla (büyük dörtlüler budanır)
36	HEX dizgisinde "0x" hiç dörtlü (4 bitlik hex hanesi) yok
37	Fonksiyon yok - çalıştırılacak birşey yok
38	Başlatılmamış bir değişken kullanılıyor
41	Fonksiyonun gövdesi yok ve çağrılmıyor
43	Tip dönüşümü sırasında muhtemel veri kaybı. Örnek: int x = (double) z;
44	Bir sabitin dönüşümü sırasında (veride) kesinlik kaybı. Örneğin: int x = M_PI
45	Karşılaştırma işleminde, işlenenlerin işaretleri farklı. Örnek: (char) c1> (uchar) c2
46	Fonksiyonun içe aktarımında problem - #import direktifi gerekli veya fonksiyonların içe aktarım özelliği kapalı
47	Açıklama çok uzun - fazla karakterler, çalıştırılabilir dosya içine eklenmeyecek

Kod	Açıklama
48	Bildirilen gösterge tamponlarının sayısı, gerekenden az
49	Göstergedeki grafiksel seriyi çizmek için hiç renk yok
50	Göstergeyi çizmek için hiç grafik serisi yok
51	'OnStart' işleyici fonksiyonu scriptte bulunmuyor
52	'OnStart' işleyici fonksiyonu yanlış parametrelerle tanımlanmış
53	'OnStart' fonksiyonu sadece bir scriptte tanımlanabilir
54	'OnInit' fonksiyonu yanlış parametrelerle tanımlanmış
55	'OnInit' fonksiyonu scriptlerde kullanılamaz
56	'OnDeinit' fonksiyonu yanlış parametrelerle tanımlanmış
57	'OnDeinit' fonksiyonu scriptlerde kullanılamaz
58	İki 'OnCalculate' fonksiyonu tanımlanmış. Tek fiyat dizili OnCalculate() kullanılacak
59	Bir karmaşık tamsayı sabitinde taşma tespit edildi
60	Değişken, büyük olasılıkla başlatılmamış.
61	Bu bildirim, belirtilen satırda bildirilmiş yerel değişkenin ifadesini imkansız kılıyor
62	>Bu bildirim, belirtilen satırda bildirilmiş global değişkenin ifadesini imkansız kılıyor
63	Statik olarak tahsis edilmiş dizi için kullanılamaz
64	Bu değişken bildirimi ön tanımlı değişkeni gizliyor
65	İfade değeri her zaman true/false'dur
66	Bir değişkeni veya bool tipli bir ifadeyi, matematiksel işlemlerde kullanmak güvenli değildir
67	Tekil çıkarma operatörünün işaretsiz ulong tipli bir değişkene uygulanmasının sonucu tanımsızdır
68	#property version özelliğinde belirtilen versiyon, Market bölümüne yerleştirmek için uygun değil; #property version tanımlayıcısının doğru biçimi: "XXX.YYY"
69	Koşullu olarak çalıştırılacak hiçbir ifade bulunamadı
70	Olay işleyici fonksiyonun bildirimi sırasında yanlış parametre kullanımı veya geçersiz dönüş tipi
71	Aynı tipli yapılar için açık yapı dönüşümü gerektirir
72	Bu bildirim, belirtilen satırda bildirilen sınıf üyesine yapılan erişimi imkansız kılar. Erişim sadece kapsam çözünürlük işlemi :: ile mümkün olacaktır
73	İkili sabit çok büyük, yüksek değerli haneler atılacak
74	Mirasçı sınıfın yöntem parametresi farklı bir const şekillendiricisine sahip; türetik sınıf, ebeveyn sınıfı aşırı-yüklemiş

Kod	Açıklama
75	Bitsel kaydırma işlemi içinde negatif veya çok büyük bir değer bulunuyor, uygulama sonucu tanımsız
76	Fonksiyon bir değere dönüş yapmalı
77	void fonksiyon bir değere dönüş yapıyor
78	Tüm kontol yolları bir değere dönüş yapmıyor
79	Global kapsamda ifadelere izin verilmez
80	Olası hatalar için işlem önceliğini kontrol edin; önceliği belirtmek için parantez kullanın
81	İki OnCalCulate() tanımlanmış. OHLC versiyonu kullanılacak.
82	Yapı eleman içermiyor, boyut 1 bayt olarak ayarlandı
83	Fonksiyonun dönüş değeri denetlenmeli
84	Kaynak gösterge hata ayıklama amacıyla derlenmiş. Bu performansı düşürüyor. Performansı artırmak için lütfen göstergeyi yeniden derleyin
85	Dizgi içinde çok büyük bir karakter kodu mevcut, 0-65535 değerleri arasında olmalı
86	Dizgi içinde tanımlanamayan karakter
87	Gösterge penceresi özelliği (ana pencerede veya alt-pencerede görüntüleme ayarı) tanımlanmamış. #property indicator_chart_window özelliği uygulandı

Derleme Hataları

MetaEdtior 5, kullanıma hazır gömülü derleyicisi ile, derleme sırasında tespit ettiği program hatalarıyla ilgili hata mesajları gösterir. Bu hataların listesi aşağıdaki tabloda verilmiştir. Bir kaynak kodunu derleyerek, çalıştırılabilir bir koda dönüştürmek için **F7** tuşuna basın. Derleyici tarafından tanımlanan hatalar giderilmedikçe, hata içeren programlar için derleme gerçekleşmeyecektir.

Kod	Açıklama
100	Dosya okuma hatası
101	Yazma amacıyla *.EX5 dosyası açma hatası
103	Derlemeyi tamamlamak için yeterli serbest bellek yok
104	Derleyici tarafından tanınmayan boş sözdizim birimi
105	#include içinde hatalı dosya ismi
106	#include içindeki dosyaya erişim hatası (muhtemelen dosya yok)
108	#define için uygun olmayan isim
109	Bilinmeyen önişlemci komutu (geçerli olanlar #include, #define, #property, #import)
110	Derleyici tarafından bilinmeyen sembol
111	Fonksiyon uygulanamadı (tarif bulunuyor ama gövde yok)
112	Çift tırnak (") atlanmış
113	Üçgen parantez (<) veya çift tırnak (") açılması unutulmuş
114	Tek tırnak (') unutulmuş
115	Üçgen kapanış parantezi ">" unutulmuş
116	Bildirim sırasında tip belirtilmemiş
117	return operatörü yok veya uygulamanın hiçbir alanında return bulunamadı
118	Parametreler için açılış parantezi gerekli
119	EX5 yazım hatası
120	Diziye geçersiz erişim
121	Fonksiyon void tipinde değil ve return operatörü bir değere dönüş yapmalı
122	Hatalı yıkıcı bildirimi
123	iki nokta işareti ":" unutulmuş
124	Değişkenin bildirimi daha önce yapılmış
125	Aynı tanımlayıcıya sahip bir değişkenin bildirimi daha önce yapılmış
126	Değişken ismi çok uzun (> 250 karakter)
127	Aynı tanımlayıcıya sahip bir yapı, daha önce bildirilmiş

Kod	Açıklama
128	Yapı tanımlanmamış
129	Aynı isimde bir yapı üyesi daha önce tanımlanmış
130	Böyle bir yapı üyesi bulunmuyor
131	Çift parantez ihlali
132	Açılış parantezi "(" gerekli
133	Dengesiz parantezler (yok "}")
134	Derlenmesi zor (çok fazla bölümleme var, iç yığın seviyeleri aşırı yüklenmiş)
135	Okuma amaçlı dosya açma hatası
136	Kaynak dosyanın indirilebileceği yeterli bellek alanı yok
137	Değişken gerekli
138	Referans başlatılamadı
140	Atama gerekli (bildirim sırasında görünür)
141	Açılış parantezi "{" gerekli
142	Parametre sadece bir dinamik dizi olabilir
143	void tipinin kullanımı kabul edilemez
144	")" veya "]" için çiftler belirtilmemiş, yani "(" veya "[" bulunmuyor
145	"(" veya "[" için çiftler belirtilmemiş, yani ")" veya "]" bulunmuyor
146	Hatalı dizi büyüklüğü
147	Parametre sayısı çok fazla (> 64)
149	Bu simge burada olmamalı
150	İşlemin hatalı kullanımı (işlenenler hatalı)
151	void tipi ifadeye izin verilmiyor
152	Operatör gerekli
153	break operatörünün yanlış kullanımı
154	Noktalı virgül ";" gerekli
155	Virgül "," gerekli
156	Bir sınıf tipi olmalı, yapı değil
157	İfade gerekli
158	HEX içinde "HEX olmayan karakter" bulundu veya sayı çok uzun (hane sayısı > 511)
159	Dizgi-sabiti 65534 karakterden fazla

Kod	Açıklama
160	Fonksiyon tanımının burada yapılması kabul edilemez
161	Beklenmeyen program sonu
162	Yapılarda ileri bildirime izin verilmez
163	Bu isimde bir fonksiyon daha önce tanımlanmış ve başka bir dönüş tipi var
164	Bu isimde bir fonksiyon daha önce farklı bir parametre kümesi ile tanımlanmış
165	Bu isimde bir fonksiyon daha önce tanımlanmış ve uygulanmış
166	Bu çağrı için fonksiyon aşırı-yüklemesi bulunamadı
167	void tipinde dönüşe sahip bir fonksiyon, bir değere dönüş yapamaz
168	Fonksiyon tanımlanmamış
170	Değer gerekli
171	*case* ifadesinde sadece tamsayı sabitleri geçerlidir
172	Bu *switch* içerisinde *case* değeri daha önce kullanılmış
173	Tamsayı gerekli
174	#import ifadesinde dosya ismi gerekli
175	Global düzeyde ifadelere izin verilmiyor
176	Noktalı virgülden ";" önce unutulmuş parantez ")"
177	Eşitliğin sol tarafında bir değişken gerekli
178	İfadenin sonucu kullanılmamış
179	*case* içinde değişken bildirimine izin verilmiyor
180	Dizgiden sayıya gizli dönüşüm
181	Sayıdan dizgiye gizli dönüşüm
182	Aşırı-yüklenmiş bir fonksiyonun belirsiz çağrısı (bir kaç aşırı-yükleme ile uyumlu)
183	Uygun bir *if* olmadan, kural dışı *else* kullanımı
184	*switch* olmadan, geçersiz *case* veya *default* kullanımı
185	Uygun olmayan üç nokta kullanımı
186	Başlatma dizilimi, başlatılan değişkenden daha fazla elemana sahip
187	*case* için bir sabit gerekli
188	Bir sabit ifade isteniyor
189	Sabit değişkenler değiştirilemez
190	Köşeli parantezin kapatılması gerekli (dizi üyesinin bildirimi sırasında)

Kod	Açıklama
191	Sayım tanımlayıcısı daha önce tanımlanmış
192	Sayımlar erişim şekillendiricilerine sahip olamaz (const, extern, static)
193	Sayım üyesinin bildirimi daha önce yapılmış
194	Aynı isimle tanımlanmış bir değişken bulunuyor
195	Aynı isimle tanımlanmış bir yapı bulunuyor
196	Sayım üyesinin ismi gerekli
197	tamsayı ifadesi gerekli
198	Sabit ifade içinde sıfıra bölüm
199	Fonksiyonda yanlış sayıda parametre bulunuyor
200	Referans ile geçirilmiş parametre, bir değişken olmalı
201	Referans ile geçirilecek aynı tipli bir değişken gerekli
202	Sabit değişken sabit olmayan referansla geçirilemez
203	Bir pozitif tamsayı sabiti isteniyor
204	Korunan sınıf üyesine erişim başarısız oldu
205	İçe aktarım daha önce farklı bir şekilde tanımlanmış
208	Çalıştırılabilir dosya oluşturulmadı
209	Gösterge için 'OnCalculate' giriş noktası bulunamadı
210	continue işlemi sadece bir döngü içinde kullanılabilir
211	private (özel) sınıf üyesine erişim hatası
213	Yapının veya sınıfın yönteminin bildirimi yapılmamış
214	private (özel) sınıf üyesine erişim hatası
216	Nesneli yapıların kopyalanmasına izin verilmiyor
218	İndis, dizi kapsamı dışında
219	Yapı veya sınıf bildirimi içinde bir dizi başlatılmasına izin verilmiyor
220	Sınıf yapıcısı parametreye sahip olamaz
221	Sınıf yıkıcısı parametreye sahip olamaz
222	Aynı isim ve parametrelere sahip sınıf yönteminin veya yapının bildirimi, daha önce yapılmış
223	İşlenen terim gerekli
224	Aynı isme ama farklı parametrelere sahip sınıf yöntemi veya yapılar mevcut (bildirim!=uygulama)

Kod	Açıklama
225	İçe aktarılan fonksiyon tarif edilmemiş
226	ZeroMemory() is not allowed for objects with protected members or inheritance
227	Aşırı-yüklenmiş fonksiyonun belirsiz çağrısı (birden fazla aşırı-yüklemede parametreler tam olarak uyuşuyor)
228	Değişken ismi gerekli
229	Burada bir referans bildirimi yapılamaz
230	Daha önce bir sayım ismi olarak kullanılmış
232	Sınıf veya yapı gerekli
235	Dizinin silinmesi için 'delete' operatörü çağrılamıyor
236	'while' operatörü gerekli
237	'delete' operatörü, bir işaretçiye sahip olmalı
238	Bu 'switch' için 'default' zaten mevcut
239	Sözdizim hatası
240	Kaçış karakteri sadece dizgilerde olabilir ('\' ile başlar)
241	Dizi isteniyor - köşeli parantez '[' diziye uygulanmamış veya, dizi olmayan parametreler dizi gibi geçirilmiş
242	Başlatma dizilimi ile başlatılamıyor
243	İçe aktarım tanımlanmamış
244	Sözdizim ağacı içinde iyileştirici hatası
245	Çok fazla yapı bildirilmiş (programı basitleştirmeyi deneyin)
246	Parametre dönüşümüne izin verilmiyor
247	'delete' operatörünün yanlış kullanımı
248	Bir referans için işaretçi bildirimine izin verilmez
249	Bir referans için referans bildirimine izin verilmez
250	Bir işaretçi için işaretçi bildirimine izin verilmez
251	Parametre listesinde yapı bildirimine izin verilmez
252	Geçersiz tip dönüşümü işlemi
253	Bir işaretçi sadece bir sınıf veya bir yapı için bildirilebilir
256	Bildirimi yapılmamış tanımlayıcı
257	Çalıştırılabilir kod iyileştirici hatası
258	Çalıştırılabilir kod oluşturma hatası

Kod	Açıklama
260	'switch' operatörü için geçersiz ifade
261	Dizgi sabitleri havuzu aşırı doldurulmuş, programı basitleştir
262	Sayıma dönüşüm yapılamaz
263	Veriler için 'virtual' ifadesini kullanmayın (sınıf üyeleri veya yapılar)
264	Sınıfın protected (korunan) yöntemi çağrılamıyor
265	Farklı bir tipe dönüş yapan, geçersiz sanal fonksiyonlar
266	Bir yapıdan sınıfa kalıtım yapılamaz
267	Bir sınıftan bir yapıya kalıtım yapılamaz
268	Yapıcı sanal olamaz (*virtual* belirtecinin kullanımına izin verilmiyor)
269	Yapılar sanal yöntemlere sahip olmazlar
270	Fonksiyonun bir gövdesi olmalı
271	Sistem fonksiyonlarının (terminal fonksiyonları) aşırı yüklenmesine izin verilmez
272	*const* belirteci, bir yapının veya sınıfın üyesi olmayan fonksiyonlarda geçersizdir
274	Sabit yöntem içindeki sınıf üyelerinin değiştirilmesine izin verilmez
276	Uygun olmayan başlatma dizilimi
277	Parametre için bir ön tanımlı değer bulunamadı (ön tanımlı parametrelerin belirli bildirimi)
278	Geçersiz ön tanımlı parametre (bildirimde ve uygulamada farklı değerler)
279	Bir sabit nesne için, sabit olmayan bir yöntemin çağrılmasına izin verilmez
280	Üyelere erişim için bir nesne gerekli (yapı veya sınıf olmayan ifadeye yapılan işaret)
281	Daha önce bildirilmiş bir yapının ismi, bildirimde kullanılamaz
284	Yetkisiz dönüşüm (kapalı kalıtımda)
285	Yapılar ve diziler, girdi değişkeni olarak kullanılmaz
286	*const* belirteci yapıcı ve yıkıcılar için geçersizdir
287	Bir datetime için hatalı dizgi ifadesi
288	Bilinmeyen özellik (#property)
289	Hatalı özellik değeri
290	#property içinde geçersiz özellik indisi
291	Çağrı parametresi atlanmış - <func (x,)>
293	Nesne referansla geçirilmeli
294	Dizi referansla geçirilmeli

Kod	Açıklama
295	Fonksiyon, dışa aktarılabilir şekilde bildirilmeli
296	Fonksiyon, dışa aktarılabilir şekilde bildirilmemiş
297	İçe aktarılan fonksiyonların, dışa aktarımına izin verilmez
298	İçe aktarılan fonksiyonlar bu parametreye sahip olamazlar (bir işaretçinin, dinamik dizi içeren bir sınıf veya yapının, sınıfın, vb. geçirilmesine izin verilmez)
299	Bir sınıf olmalı
300	#import kapatılmamış
302	Tip uyumsuzluğu
303	Extern değişken daha önce başlatılmış
304	Hiç dışa aktarılmış fonksiyon veya giriş noktası bulunamadı
305	Açık yapı çağrısına izin verilmiyor
306	Yöntem, sabit olarak bildirilmiş
307	Yöntem, sabit olarak bildirilmemiş
308	Hatalı kaynak dosyası boyutu
309	Hatalı kaynak ismi
310	Kaynak dosyası açma hatası
311	Kaynak dosyası okuma hatası
312	Bilinmeyen kaynak tipi
313	Kaynak dosyasının, adresi hatalı
314	Belirtilen kaynak ismi daha önce kullanılmış
315	Fonksiyon benzeri makro için argüman gerekiyor
316	Makro tanımında beklenmeyen sembol
317	Makronun biçimsel parametrelerinde hata
318	Makro için geçersiz parametre sayısı
319	Parametre sayısı bir makro için çok fazla
320	Çok karmaşık, Makroyu basitleştirin
321	EnumToString() parametresi sadece bir sayım olabilir
322	Kaynak ismi çok uzun
323	Desteklenmeyen görüntü biçimi (sadece 24 veya 32 bitlik renk derinliğine sahip BMP biçimi desteklenir)
324	Dizilerin bildirimi operatör içinde yapılamaz

Kod	Açıklama
325	Fonksiyonun bildirimi sadece global düzeyde yapılabilir
326	Bu kapsam için bildirime izin verilmiyor
327	Statik değişkenlerin, yerel değişkenlerin değerleri ile başlatılmasına izin verilmez
328	Ön tanımlı bir yapıcıya sahip olmayan bir nesne dizisinin kural dışı bildirimi
329	Başlatma listesine, sadece yapıcılar için izin verilir
330	Başlatma listesinden sonra fonksiyon tanımı bulunmuyor
331	Başlatma listesi boş
332	Yapıcı içinde, bir dizinin başlatılmasına izin verilmiyor
333	Başlatma listesinde ebeveyn sınıfın üyelerinin başlatılmasına izin verilmiyor
334	Tamsayı tipli ifade gerekli
335	Dizi için istenen bellek, maksimum değerin üzerinde
336	Yapı için istenen bellek, maksimum değerin üzerinde
337	Global düzeyde bildirimi yapılan değişken için istenen bellek, maksimum değerin üzerinde
338	Yerel değişkenler için istenen bellek, maksimum değerin üzerinde
339	Yapıcı tanımlanmamış
340	Geçersiz ikon dosyası ismi
341	İkon dosyası belirtilen konumda açılamadı
342	İkon dosyası yanlış ve ICOformatında olmalıdır.
343	Başlatma listesi kullanılarak bir yapının/sınıfın yapıcısındaki bir üyenin yeniden başlatılması
344	Yapıcının başlatma listesindeki statik üyelerin, başlatılmasına izin verilmiyor
345	Bir sınıfın/yapının statik olmayan üyesinin global düzeyde başlatılmasına izin verilmiyor
346	Yapı/sınıf yönteminin ismi, daha önce bildirilmiş bir üyenin ismiyle örtüşüyor
347	Yapı/sınıf üyesinin ismi, daha önce bildirilmiş bir yöntemin ismiyle örtüşüyor
348	Sanal fonksiyon, static şekline bildirilemez
349	const şekillendiricisine statik fonksiyonlarda izin verilmez
350	Yapıcı veya yıkıcı, statik olamaz
351	Bir sınıfın veya yapının statik olmayan üyesine/yöntemine statik fonksiyon gibi erişilemez

Kod	Açıklama
352	Bir aşırı-yükleme işleminin (+,-,[],++,-- vb.) operator anahtar sözcüğünden sonra gelmesi beklenir
353	MQL5 içinde tüm işlemler aşırı-yüklenmiş olamaz
354	Tanımlama bildirimle uyuşmuyor
355	operatör için geçersiz sayıda parametre belirtilmiş
356	Olay işleme fonksiyonu bulunamadı
357	Yöntem dışarı aktarılamaz
358	Sabit nesneye yapılan bir işaretçi, sabit olmayan bir nesne ile normalleştirilemez
359	Sınıf şablonları henüz desteklenmiyor
360	Fonksiyon şablonunun aşırı-yüklenmesi henüz desteklenmiyor
361	Fonksiyon şablonu uygulanamaz
362	Fonksiyon şablonunda belirsiz parametre (birkaç parametre tipi uygulanmış olabilir)
363	Fonksiyon şablonunu argümanını normalleştirecek parametrenin tipi belirlenemiyor
364	Fonksiyon şablonunda, hatalı parametre sayısı
365	Fonksiyon şablonu sanal olamaz
366	Fonksiyon şablonu dışa aktarılamaz
367	Fonksiyon şablonu içe aktarılamaz
368	Nesne içeren yapılara izin verilmez
369	Nesneler içeren dizge dizilerine ve yapılara izin verilmez
370	Statik bir sınıf/yapı üyesi açıkça başlatılmalıdır
371	Derleyici sınırlaması: dizge en fazla 65.535 karakter içerebilir
372	Tutarsız #ifdef/#endif
373	Sınıfın nesnesi geri döndürülemez, kopya yapıcısı bulunamadı
374	Statik olmayan üyeler ve yöntemler kullanılamaz
375	OnTesterInit(), OnTesterDeinit() olmadan kullanılması imkansızdır
376	'%s' resmi parametrenin yeniden tanımlaması
377	__FUNCSIG__ ve __FUNCTION__ makroları, fonksiyon gövdesinin dışında görünemez
378	Geçersiz geri dönüş tipi. Bu hata, örneğin, yapı veya işaretçi geri döndüren DLL'den içe aktarılan fonksiyonlar için oluşturulacaktır.
379	Şablon kullanım hatası
380	Kullanılmıyor

Kod	Açıklama
381	Sanal fonksiyon bildiriminde hatalı sözdizim, sadece "=NULL" veya "=0" kullanımına izin verilir
382	Sadece sanal fonksiyonların bildirimi soyut belirteçlerle ("=NULL" veya "=0") yapılabilir
383	Soyut sınıf örneği oluşturulamaz
384	Kullanıcı tanımlı bir tipin işaretçisi, dynamic_cast operatörü kullanılarak dinamik dönüşüm için hedef tip olarak uygulanmalıdır.
385	"Fonksiyon İşaretçisi" tipi beklenmektedir
386	Metot işaretçileri desteklenmemektedir
387	Hata - fonksiyon işaretçisinin tipi tanımlanamamaktadır
388	Özel kalıtım nedeniyle dönüştürme kullanılamaz
389	const değiştiricisi içeren bir değişken bildirim sırasında başlatılmalıdır
393	Arayüzde sadece public erişimi olan metotlar bildirilebilir
394	Bir arayüzün geçersiz bir şekilde başka bir arayüzün içine yerleştirilmesi
395	Bir arayüz sadece başka bir arayüzden elde edilebilir
396	Bir arayüz beklenmektedir
397	Arayüzler sadece genele açık kalıtımı destekler
398	Bir arayüz üye içeremez
399	Arayüz nesneleri doğrudan oluşturulamaz, yalnızca kalıtım kullanılarak oluşturulabilir
400	Bir belirteç bir ileri bildirimde kullanılamaz
401	final belirteci ile bildirildiği için sınıftan kalıtım imkansızdır.
402	final belirteci ile bildirilen bir metot yeniden tanımlanamamaktadır
403	final belirteci yalnızca sanal fonksiyonlara uygulanabilir
404	override belirteci tarafından işaretlenen metot aslında herhangi bir temel sınıf fonksiyonunu geçersiz kılmaz
405	Bir fonksiyon tanımlanırken belirtece izin verilmez, yalnızca bildirilirken izin verilir
406	Tip belirtilene dönüştürülememektedir
407	Tip bir kaynak değişkeni için kullanılamaz
408	Proje dosyasında hata
409	Birleşim üyesi olarak kullanılamaz
410	Belirsiz ad seçimi, kullanım bağlamı açıkça tanımlanmalıdır
411	Yapı DLL'den kullanılamaz

Kod	Açıklama
412	delete belirteci tarafından işaretlenmiş bir fonksiyon çağrılamamaktadır
413	MQL4 desteklenmemektedir. Programı derlemek için MetaTrader 4 terminal kurulum klasöründeki MetaEditor'ı kullanın

Çalışma Zamanı Hataları

GetLastError(), ön tanımlı _LastError değişkenindeki son hata koduna dönüş yapan bir fonksiyondur. Bu değer, ResetLastError() fonksiyonu kullanılarak sıfırlanabilir.

Sabit	Kod	Açıklama
ERR_SUCCESS	0	İşlem başarıyla tamamlandı
ERR_INTERNAL_ERROR	4001	Beklenmeyen içsel hata
ERR_WRONG_INTERNAL_PARAMETER	4002	Müşteri terminali fonksiyonunun çağrısında yanlış parametre kullanımı
ERR_INVALID_PARAMETER	4003	Sistem fonksiyonunun çağrısında yanlış fonksiyon kullanımı
ERR_NOT_ENOUGH_MEMORY	4004	Sistem fonksiyonunu uygulamak için yeterli bellek yok
ERR_STRUCT_WITHOBJECTS_ORCLASS	4005	Yapı, dizgi nesnelerini ve/veya dinamik dizileri ve/veya bu tip nesnelerin yapılarını ve/veya sınıflarını içeriyor
ERR_INVALID_ARRAY	4006	Yanlış tipli veya yanlış büyüklükte bir dizi veya hasarlı bir nesne veya bir dinamik dizi
ERR_ARRAY_RESIZE_ERROR	4007	Dizinin yer değişimi için yeterli belek yok, veya bir statik dizinin büyüklüğünü değiştirme girişimi yapıldı
ERR_STRING_RESIZE_ERROR	4008	Dizginin yeniden yerleştirilmesi için yeterli bellek yok
ERR_NOTINITIALIZED_STRING	4009	Başlatılmamış dizgi
ERR_INVALID_DATETIME	4010	Geçersiz tarih ve/veya zaman
ERR_ARRAY_BAD_SIZE	4011	Dizideki elemanların sayısı 2147483647'yi geçemez
ERR_INVALID_POINTER	4012	Yanlış işaretçi
ERR_INVALID_POINTER_TYPE	4013	Yanlış tipte işaretçi
ERR_FUNCTION_NOT_ALLOWED	4014	Sistem fonksiyonunun çağrılmasına izin verilmiyor
ERR_RESOURCE_NAME_DUPLICATED	4015	dynamic ve static kaynakların isimleri uyuşuyor
ERR_RESOURCE_NOT_FOUND	4016	EX5 içinde bu isimde kaynak bulunamadı

Sabit	Kod	Açıklama
ERR_RESOURCE_UNSUPPORTED_TYPE	4017	Desteklenmeyen kaynak tipi veya kaynağın boyutu 16 Mb'i geçiyor
ERR_RESOURCE_NAME_IS_TOO_LONG	4018	kaynak ismi 63 karakteri aşıyor
ERR_MATH_OVERFLOW	4019	Matematiksel fonksiyon hesaplanırken taşma oluştu
ERR_SLEEP_ERROR	4020	Çağrılan Sleep(), test bitiş zamanını aşıyor
ERR_PROGRAM_STOPPED	4022	Test zorla dışarıdan durduruldu. Örneğin, optimizasyon kesintiye uğratıldı, görsel test penceresi kapatıldı veya test temsilcisi durduruldu
ERR_INVALID_TYPE	4023	Geçersiz tür
ERR_INVALID_HANDLE	4024	Geçersiz tanıtıcı
ERR_TOO_MANY_OBJECTS	4025	Nesne havuzu doldu
Çizelgeler		
ERR_CHART_WRONG_ID	4101	Yanlış çizelge kimliği (tanımlayıcısı)
ERR_CHART_NO_REPLY	4102	Çizelge yanıt vermiyor
ERR_CHART_NOT_FOUND	4103	Çizelge bulunamadı
ERR_CHART_NO_EXPERT	4104	Çizelgede, olayı işleyebilecek bir Uzman Danışman yok
ERR_CHART_CANNOT_OPEN	4105	Çizelge açılma hatası
ERR_CHART_CANNOT_CHANGE	4106	Çizelge periyodunu ve sembolünü değiştirme işlemi başarısız oldu
ERR_CHART_WRONG_PARAMETER	4107	Çizelge ile çalışma fonksiyonu için kullanılan parametrenin hata değeri
ERR_CHART_CANNOT_CREATE_TIMER	4108	Sayacı başlatma başarısız
ERR_CHART_WRONG_PROPERTY	4109	Yanlış çizelge özelliği tanımlayıcısı
ERR_CHART_SCREENSHOT_FAILED	4110	Ekran görüntüsü oluşturma hatası
ERR_CHART_NAVIGATE_FAILED	4111	Çizelge boyunca konumlandırma hatası
ERR_CHART_TEMPLATE_FAILED	4112	Şablon uygulama hatası
ERR_CHART_WINDOW_NOT_FOUND	4113	Göstergeyi içeren alt pencere bulunamadı

Sabit	Kod	Açıklama
ERR_CHART_INDICATOR_CANNOT_ADD	4114	Çizelgeye gösterge eklerken hata oluştu
ERR_CHART_INDICATOR_CANNOT_DEL	4115	Göstergeyi çizelgeden silerken hata oluştu
ERR_CHART_INDICATOR_NOT_FOUND	4116	Gösterge belirtilen çizelge üzerinde bulunamadı
Grafiksel Nesneler		
ERR_OBJECT_ERROR	4201	Grafiksel nesne ile çalışma hatası
ERR_OBJECT_NOT_FOUND	4202	Grafiksel nesne bulunamadı
ERR_OBJECT_WRONG_PROPERTY	4203	Grafiksel nesne için yanlış tanımlayıcı
ERR_OBJECT_GETDATE_FAILED	4204	Değere karşılık gelen tarih alınamadı
ERR_OBJECT_GETVALUE_FAILED	4205	Tarihe karşılık gelen değer alınamadı
MarketInfo		
ERR_MARKET_UNKNOWN_SYMBOL	4301	Bilinmeyen sembol
ERR_MARKET_NOT_SELECTED	4302	Sembol, Piyasa Gözleminde seçilmedi
ERR_MARKET_WRONG_PROPERTY	4303	Sembol özelliği için yanlış tanımlayıcı
ERR_MARKET_LASTTIME_UNKNOWN	4304	Son tik zamanı bilinmiyor (hiç tik yok)
ERR_MARKET_SELECT_ERROR	4305	Sembolün Piyasa Gözleminden silinmesinde hata
Geçmişe Erişimi		
ERR_HISTORY_NOT_FOUND	4401	İstenen geçmiş bulunamadı
ERR_HISTORY_WRONG_PROPERTY	4402	Geçmiş özeliğinin tanımlayıcısı yanlış
ERR_HISTORY_TIMEOUT	4403	Geçmiş veri isteğinde zamanaşımı oluştu
ERR_HISTORY_BARS_LIMIT	4404	İstenen çubuk sayısı terminal ayarlarında sınırlandırılmış
ERR_HISTORY_LOAD_ERRORS	4405	Geçmiş veri yüklemesi sırasında hatalar oluştu

Sabit	Kod	Açıklama
ERR_HISTORY_SMALL_BUFFER	4407	Hedef dizi, istenen veriyi depolamak için çok küçük
Global Değişkenler		
ERR_GLOBALVARIABLE_NOT_FOUND	4501	Müşteri terminalinin global değişkeni bulunamadı
ERR_GLOBALVARIABLE_EXISTS	4502	Aynı isimde bir müşteri terminali global değişkeni mevcut
ERR_GLOBALVARIABLE_NOT_MODIFIED	4503	Global değişkenler ayarlanmamış
ERR_GLOBALVARIABLE_CANNOTREAD	4504	Global değişken değerlerini içeren dosya okunamadı
ERR_GLOBALVARIABLE_CANNOTWRITE	4505	Global değişken değerlerini içeren dosyaya yazılamadı
ERR_MAIL_SEND_FAILED	4510	Email gönderimi başarısız oldu
ERR_PLAY_SOUND_FAILED	4511	Ses oynatımı başarısız oldu
ERR_MQL5_WRONG_PROPERTY	4512	Program özelliğinin tanımlayıcısı yanlış
ERR_TERMINAL_WRONG_PROPERTY	4513	Terminal özelliğinin tanımlayıcısı yanlış
ERR_FTP_SEND_FAILED	4514	Ftp ile dosya gönderimi başarısız oldu
ERR_NOTIFICATION_SEND_FAILED	4515	Uyarı gönderme başarısız oldu
ERR_NOTIFICATION_WRONG_PARAMETER	4516	Uyarı gönderimi için geçersiz parametre - boş bir dizgi veya NULL değeri SendNotification() fonksiyonuna geçirilmiş
ERR_NOTIFICATION_WRONG_SETTINGS	4517	Terminalin uyarı ayarları hatalı yapılmış (tanımlayıcı belirtilmemiş veya izin durumu ayarlanmamış)
ERR_NOTIFICATION_TOO_FREQUENT	4518	Uyarı gönderimi çok sık
Özel Gösterge Tamponları		
ERR_BUFFERS_NO_MEMORY	4601	Gösterge tamponlarının dağıtılması için yeterli bellek yok
ERR_BUFFERS_WRONG_INDEX	4602	Gösterge tamponunun indisi yanlış
Özel Gösterge Özellikleri		
ERR_CUSTOM_WRONG_PROPERTY	4603	Özel gösterge özeliğinin tanımlayıcısı yanlış

Sabit	Kod	Açıklama
Hesap		
ERR_ACCOUNT_WRONG_PROPERTY	4701	Hesap özelliğinin tanımlayıcısı yanlış
ERR_TRADE_WRONG_PROPERTY	4751	Alım-satım özelliği tanımlayıcısı yanlış
ERR_TRADE_DISABLED	4752	Uzman Danışmanlar ile alım-satım yapmak yasaklanmış
ERR_TRADE_POSITION_NOT_FOUND	4753	Pozisyon bulunamadı
ERR_TRADE_ORDER_NOT_FOUND	4754	Emir bulunamadı
ERR_TRADE_DEAL_NOT_FOUND	4755	İşlem bulunamadı
ERR_TRADE_SEND_FAILED	4756	Alım-satım isteğinin gönderilmesi başarısız oldu
ERR_TRADE_CALC_FAILED	4758	Kar veya teminat değeri hesaplanamadı
Göstergeler		
ERR_INDICATOR_UNKNOWN_SYMBOL	4801	Bilinmeyen sembol
ERR_INDICATOR_CANNOT_CREATE	4802	Gösterge oluşturulamıyor
ERR_INDICATOR_NO_MEMORY	4803	Göstergeyi eklemek için yeterli bellek yok
ERR_INDICATOR_CANNOT_APPLY	4804	Bu gösterge başka bir göstergeye uygulanamaz
ERR_INDICATOR_CANNOT_ADD	4805	Göstergenin çizelgeye uygulanmasında hata
ERR_INDICATOR_DATA_NOT_FOUND	4806	İstenen veri bulunamadı
ERR_INDICATOR_WRONG_HANDLE	4807	Göstergenin tanıtıcı değeri yanlış
ERR_INDICATOR_WRONG_PARAMETERS	4808	Göstergedeki parametre sayısı yanlış
ERR_INDICATOR_PARAMETERS_MISSING	4809	Göstergede hiç parametre yok
ERR_INDICATOR_CUSTOM_NAME	4810	Dizideki ilk parametre, özel göstergenin ismi olmalı
ERR_INDICATOR_PARAMETER_TYPE	4811	Gösterge oluşturulması sırasında, dizi içinde geçersiz parametre tipi
ERR_INDICATOR_WRONG_INDEX	4812	İstenen gösterge tamponunun indisi hatalı
Piyasa Derinliği		

Sabit	Kod	Açıklama
ERR_BOOKS_CANNOT_ADD	4901	Piyasa Derinliği eklenemiyor
ERR_BOOKS_CANNOT_DELETE	4902	Piyasa Derinliği kaldırılamıyor
ERR_BOOKS_CANNOT_GET	4903	Piyasa Derinliğinden veri alınamıyor
ERR_BOOKS_CANNOT_SUBSCRIBE	4904	Piyasa Derinliğinden veri alımı için abonelik hatası
Dosya İşlemleri		
ERR_TOO_MANY_FILES	5001	64 dosyadan fazlası aynı anda açılamaz
ERR_WRONG_FILENAME	5002	Geçersiz dosya ismi
ERR_TOO_LONG_FILENAME	5003	Dosya ismi çok uzun
ERR_CANNOT_OPEN_FILE	5004	Dosya açma hatası
ERR_FILE_CACHEBUFFER_ERROR	5005	Okuma için yeterli önbellek alanı yok
ERR_CANNOT_DELETE_FILE	5006	Dosya silme hatası
ERR_INVALID_FILEHANDLE	5007	Bu tanıtıcı değere sahip bir dosya kapatıldı, veya hiç açılmadı
ERR_WRONG_FILEHANDLE	5008	Dosya için yanlış tanıtıcı değer
ERR_FILE_NOTTOWRITE	5009	Dosya yazım amacıyla açılmalıdır
ERR_FILE_NOTTOREAD	5010	Dosya okuma amacıyla açılmalıdır
ERR_FILE_NOTBIN	5011	Dosya, ikili (binary) dosya şeklinde açılmalıdır
ERR_FILE_NOTTXT	5012	Dosya, metin (txt) dosyası şeklinde açılmalıdır
ERR_FILE_NOTTXTORCSV	5013	Dosya, metin (txt) veya CSV dosyası şeklinde açılmalıdır
ERR_FILE_NOTCSV	5014	Dosya, CSV dosyası şeklinde açılmalıdır
ERR_FILE_READERROR	5015	Dosya okuma hatası
ERR_FILE_BINSTRINGSIZE	5016	Dosya ikili biçimde açıldığı için dizgi büyüklüğü belirtilmelidir
ERR_INCOMPATIBLE_FILE	5017	Metin dosyaları dizgi dizileri için olmalıdır, diğer diziler için ikili (binary) dosyalar kullanılmalıdır
ERR_FILE_IS_DIRECTORY	5018	Bu bir dosya değil, bir klasör

Sabit	Kod	Açıklama
ERR_FILE_NOT_EXIST	5019	Dosya mevcut değil
ERR_FILE_CANNOT_REWRITE	5020	Dosya yeniden yazılamaz
ERR_WRONG_DIRECTORYNAME	5021	Dosya yolunun ismi yanlış
ERR_DIRECTORY_NOT_EXIST	5022	Böyle bir dosya yolu bulunmuyor
ERR_FILE_ISNOT_DIRECTORY	5023	Bu bir dosya, klasör değil
ERR_CANNOT_DELETE_DIRECTORY	5024	Klasör kaldırılamaz
ERR_CANNOT_CLEAN_DIRECTORY	5025	Klasörün kaldırılması başarısız oldu (büyük olasılıkla bir veya iki dosya bloke edilmiş ve kaldırma işlemi başarısız olmuş)
ERR_FILE_WRITEERROR	5026	Kaynağı dosyaya yazma işlemi başarısız oldu
ERR_FILE_ENDOFFILE	5027	Verinin sonraki kısımları CSV dosyasından okunamıyor (FileReadString, FileReadNumber, FileReadDatetime, FileReadBool) dosyanın sonuna ulaşıldı
Dizgi Dönüşümü		
ERR_NO_STRING_DATE	5030	Dizgide tarih yok
ERR_WRONG_STRING_DATE	5031	Dizgideki tarih yanlış
ERR_WRONG_STRING_TIME	5032	Dizgideki zaman yanlış
ERR_STRING_TIME_ERROR	5033	Dizgiyi tarihe çevirme hatası
ERR_STRING_OUT_OF_MEMORY	5034	Dizgi için yeterli bellek yok
ERR_STRING_SMALL_LEN	5035	Dizgi uzunluğu beklenenden az
ERR_STRING_TOO_BIGNUMBER	5036	Sayı çok büyük, ULONG_MAX değerinden fazla
ERR_WRONG_FORMATSTRING	5037	Geçersiz biçim dizgisi
ERR_TOO_MANY_FORMATTERS	5038	Biçim belirteçlerinin sayısı, parametrelerden fazla
ERR_TOO_MANY_PARAMETERS	5039	Parametrelerin sayısı, biçim belirteçlerinden fazla
ERR_WRONG_STRING_PARAMETER	5040	string tipli hatalı parametre
ERR_STRINGPOS_OUTOFRANGE	5041	Konum, dizginin dışında
ERR_STRING_ZEROADDED	5042	Dizginin sonuna 0 eklenmiş, kullanışsız işlem

Sabit	Kod	Açıklama
ERR_STRING_UNKNOWNTYPE	5043	Dizgiye dönüşüm yaparken bilinmeyen veri tipi
ERR_WRONG_STRING_OBJECT	5044	Hasarlı dizgi nesnesi
Dizilerle İşlemler		
ERR_INCOMPATIBLE_ARRAYS	5050	Uyumlu olmayan dizilerin kopyalanması. Dizgi dizisi sadece bir dizgi dizisine kopyalanabilir, aynı şekilde bir sayısal dizi de sadece sayısal diziye kopyalanabilir
ERR_SMALL_ASSERIES_ARRAY	5051	Alınan dizi AS_SERIES şeklinde bildirilmiş ve yeterli büyüklükte değil
ERR_SMALL_ARRAY	5052	Dizi çok küçük, başlangıç konumu dizinin dışında
ERR_ZEROSIZE_ARRAY	5053	Dizinin büyüklüğü sıfır
ERR_NUMBER_ARRAYS_ONLY	5054	Nümerik bir dizi olmalı
ERR_ONEDIM_ARRAYS_ONLY	5055	Tek boyutlu bir dizi olmalı
ERR_SERIES_ARRAY	5056	Zaman serileri kullanılamaz
ERR_DOUBLE_ARRAY_ONLY	5057	double tipli bir dizi olmalı
ERR_FLOAT_ARRAY_ONLY	5058	float tipli bir dizi olmalı
ERR_LONG_ARRAY_ONLY	5059	long tipli bir dizi olmalı
ERR_INT_ARRAY_ONLY	5060	int tipli bir dizi olmalı
ERR_SHORT_ARRAY_ONLY	5061	short tipli bir dizi olmalı
ERR_CHAR_ARRAY_ONLY	5062	char tipli bir dizi olmalı
ERR_STRING_ARRAY_ONLY	5063	Sadece dizgi dizileri
OpenCL İşlemleri		
ERR_OPENCL_NOT_SUPPORTED	5100	OpenCL fonksiyonları bu bilgisayarda desteklenmiyor
ERR_OPENCL_INTERNAL	5101	OpenCL çalışması sırasında bir hata oluştu
ERR_OPENCL_INVALID_HANDLE	5102	Geçersiz OpenCL işleyici
ERR_OPENCL_CONTEXT_CREATE	5103	OpenCL bağlamı oluşturma hatası
ERR_OPENCL_QUEUE_CREATE	5104	OpenCL içinde bir çalıştırma kuyruğu oluşturma hatası

Sabit	Kod	Açıklama
ERR_OPENCL_PROGRAM_CREATE	5105	OpenCL programının derlenmesi sırasında hata oluştu
ERR_OPENCL_TOO_LONG_KERNEL_NAME	5106	Kernel ismi çok uzun (OpenCL kernel)
ERR_OPENCL_KERNEL_CREATE	5107	OpenCL kerneli oluşturma hatası
ERR_OPENCL_SET_KERNEL_PARAMETER	5108	OpenCL kerneli için parametrelerin ayarlanması sırasında hata oluştu
ERR_OPENCL_EXECUTE	5109	OpenCL programı çalışma zamanı hatası
ERR_OPENCL_WRONG_BUFFER_SIZE	5110	OpenCL tamponu için geçersiz büyüklük
ERR_OPENCL_WRONG_BUFFER_OFFSET	5111	OpenCL tamponu içinde geçersiz konum
ERR_OPENCL_BUFFER_CREATE	5112	OpenCL tamponu oluşturma hatası
ERR_OPENCL_TOO_MANY_OBJECTS	5113	Çok fazla OpenCL nesnesi var
ERR_OPENCL_SELECTDEVICE	5114	OpenCL cihaz seçimi hatası
Veritabanlarıyla çalışma		
ERR_DATABASE_INTERNAL	5120	Dahili veritabanı hatası
ERR_DATABASE_INVALID_HANDLE	5121	Geçersiz veritabanı tanıtıcısı
ERR_DATABASE_TOO_MANY_OBJECTS	5122	Kabul edilebilir maksimum Database nesnesi sayısı aşıldı
ERR_DATABASE_CONNECT	5123	Veritabanı bağlantı hatası
ERR_DATABASE_EXECUTE	5124	İstek yürütme hatası
ERR_DATABASE_PREPARE	5125	İstek oluşturma hatası
ERR_DATABASE_NO_MORE_DATA	5126	Okunacak daha fazla veri yok
ERR_DATABASE_STEP	5127	Bir sonraki istek girişine geçerken hata oluştu
ERR_DATABASE_NOT_READY	5128	İstek sonuçlarını okumak için veriler henüz hazır değil
ERR_DATABASE_BIND_PARAMETERS	5129	SQL isteğinde parametrelerin otomatik değiştirilmesi başarısız oldu
WebRequest işlemleri		
ERR_WEBREQUEST_INVALID_ADDRESS	5200	Geçersiz URL

Sabit	Kod	Açıklama
ERR_WEBREQUEST_CONNECT_FAILED	5201	Belirtilen URL ile bağlantı kurulamadı
ERR_WEBREQUEST_TIMEOUT	5202	Zaman-aşımı oluştu
ERR_WEBREQUEST_REQUEST_FAILED	5203	HTTP isteği başarısız
Ağ ile İşlemler (soketler)		
ERR_NETSOCKET_INVALIDHANDLE	5270	Fonksiyona geçersiz soket tanıtıcısı aktarıldı
ERR_NETSOCKET_TOO_MANY_OPENED	5271	Çok fazla açık soket(maksimum 128)
ERR_NETSOCKET_CANNOT_CONNECT	5272	Uzak ana bilgisayara bağlanılamadı
ERR_NETSOCKET_IO_ERROR	5273	Soketten veri gönderilemedi/alınamadı
ERR_NETSOCKET_HANDSHAKE_FAILED	5274	Güvenli bağlantı kurulamadı (TLS Tokalaşma)
ERR_NETSOCKET_NO_CERTIFICATE	5275	Bağlantıyı koruyan sertifikada veri yok
Özel semboller		
ERR_NOT_CUSTOM_SYMBOL	5300	Kullanıcı-tanımlı bir sembol belirtilmeli
ERR_CUSTOM_SYMBOL_WRONG_NAME	5301	Kullanıcı-tanımlı sembolün ismi geçersiz. Sembol ismi sadece Latin karakterleri ve ".", "_", "&" ve "#" karakterlerini içerebilir, bunların dışındaki noktalama işaretlerini, boşluk ve diğer özel karakterleri içeremez. <, >, :, ", / ,\, |, ?, * karakterlerinin kullanılması önerilmez.
ERR_CUSTOM_SYMBOL_NAME_LONG	5302	Kullanıcı-tanımlı sembolün ismi çok uzun. Sembol ismi, sondaki 0 karakteri de dahil olmak üzere 32 karakteri geçemez
ERR_CUSTOM_SYMBOL_PATH_LONG	5303	Kullanıcı-tanımlı sembolün adresi çok uzun. Adres; "Custom\\", sembol ismi, grup ayraçları ve sondaki 0 karakteri de dahil olmak üzere 128 karakterden fazla olamaz

Sabit	Kod	Açıklama
ERR_CUSTOM_SYMBOL_EXIST	5304	Aynı isimde başka bir sembol mevcut
ERR_CUSTOM_SYMBOL_ERROR	5305	Kullanıcı-tanımlı sembolün oluşturulması, silinmesi veya değiştirilmesi sırasında hata meydana geldi
ERR_CUSTOM_SYMBOL_SELECTED	5306	Piyasa gözleminde seçili olan kullanıcı-tanımlı bir sembolü silmeyi deniyorsunuz
ERR_CUSTOM_SYMBOL_PROPERTY_WRONG	5307	Kullanıcı-tanımlı sembol için geçersiz özellik
ERR_CUSTOM_SYMBOL_PARAMETER_ERROR	5308	Kullanıcı-tanımlı sembol özelliği ayarlanırken hatalı parametre kullanıldı
ERR_CUSTOM_SYMBOL_PARAMETER_LONG	5309	Kullanıcı-tanımlı sembol özelliğinin parametresi için kullanılan dizgi çok uzun
ERR_CUSTOM_TICKS_WRONG_ORDER	5310	Dizideki tikler zamana göre sıralanmamıştır
Ekonomik Takvim		
ERR_CALENDAR_MORE_DATA	5400	Dizi boyutu, tüm değerlerin açıklamalarını elde etmek için yetersizdir
ERR_CALENDAR_TIMEOUT	5401	İstek süresi sınırı aşıldı
ERR_CALENDAR_NO_DATA	5402	Ülke bulunamadı
Veritabanlarıyla çalışma		
ERR_DATABASE_ERROR	5601	Genel hata
ERR_DATABASE_LOGIC	5602	SQLite dahili mantık hatası
ERR_DATABASE_PERM	5603	Erişim reddedildi
ERR_DATABASE_ABORT	5604	Geri arama prosedürü durdurma istedi
ERR_DATABASE_BUSY	5605	Veritabanı dosyası kilitli
ERR_DATABASE_LOCKED	5606	Veritabanı tablosu kilitli
ERR_DATABASE_NOMEM	5607	İşlemi tamamlamak için yetersiz bellek
ERR_DATABASE_READONLY	5608	Salt okunur veritabanına yazma girişimi

Sabit	Kod	Açıklama
ERR_DATABASE_INTERRUPT	5609	İşlem sqlite3_interrupt() tarafından sonlandırıldı
ERR_DATABASE_IOERR	5610	Disk girdi-çıktı hatası
ERR_DATABASE_CORRUPT	5611	Veritabanı disk görüntüsü bozuk
ERR_DATABASE_NOTFOUND	5612	Sqlite3_file_control()'de bilinmeyen işlem kodu
ERR_DATABASE_FULL	5613	Veritabanı dolu olduğu için ekleme başarısız oldu
ERR_DATABASE_CANTOPEN	5614	Veritabanı dosyası açılamıyor
ERR_DATABASE_PROTOCOL	5615	Veritabanı kilit protokolü hatası
ERR_DATABASE_EMPTY	5616	Sadece dahili kullanım içindir
ERR_DATABASE_SCHEMA	5617	Veritabanı şeması değişti
ERR_DATABASE_TOOBIG	5618	Dizge veya BLOB boyut sınırını aşıyor
ERR_DATABASE_CONSTRAINT	5619	Kısıtlama ihlali nedeniyle durduruldu
ERR_DATABASE_MISMATCH	5620	Veri tipi uyumsuzluğu
ERR_DATABASE_MISUSE	5621	Kütüphanenin yanlış kullanımı
ERR_DATABASE_NOLFS	5622	Kullanılan işletim sistemi işlevleri ana bilgisayar tarafından desteklenmiyor
ERR_DATABASE_AUTH	5623	Yetkilendirme reddedildi
ERR_DATABASE_FORMAT	5624	Kullanılmıyor
ERR_DATABASE_RANGE	5625	Bağlama parametresi hatası, yanlış indeks
ERR_DATABASE_NOTADB	5626	Açılan dosya bir veritabanı dosyası değil
Matris ve Vektör Metotları		
ERR_MATRIX_INTERNAL	5700	Matris/vektör yürütme alt sisteminin iç hatası
ERR_MATRIX_NOT_INITIALIZED	5701	Matris/vektör başlatılamadı
ERR_MATRIX_INCONSISTENT	5702	İşlemdeki matrislerin/vektörlerin boyutları tutarsız
ERR_MATRIX_INVALID_SIZE	5703	Geçersiz matris/vektör büyüklüğü
ERR_MATRIX_INVALID_TYPE	5704	Geçersiz matris/vektör türü

Sabit	Kod	Açıklama
ERR_MATRIX_FUNC_NOT_ALLOWED	5705	Bu matris/vektör için fonksiyon mevcut değil
ERR_MATRIX_CONTAINS_NAN	5706	Matris/vektör sayısal olmayan değer (Nan/Inf) içeriyor
ONNX Modelleri		
ERR_ONNX_INTERNAL	5800	ONNX iç hatası
ERR_ONNX_NOT_INITIALIZED	5801	ONNX Runtime API başlatma hatası
ERR_ONNX_NOT_SUPPORTED	5802	MQL5 tarafından desteklenmeyen özellik veya değer
ERR_ONNX_RUN_FAILED	5803	ONNX Runtime API çalıştırma hatası
ERR_ONNX_INVALID_PARAMETERS_COUNT	5804	OnnxRun'a geçersiz sayıda parametre iletildi
ERR_ONNX_INVALID_PARAMETER	5805	Geçersiz parametre değeri
ERR_ONNX_INVALID_PARAMETER_TYPE	5806	Geçersiz parametre türü
ERR_ONNX_INVALID_PARAMETER_SIZE	5807	Geçersiz parametre büyüklüğü
ERR_ONNX_WRONG_DIMENSION	5808	Tensör boyutu ayarlanmamış veya geçersiz
ERR_USER_ERROR_FIRST	65536	Kullanıcı tanımlı hatalar bu kodla başlar
Kullanıcı-Tanımlı Hatalar		
ERR_USER_ERROR_FIRST	65536	Kullanıcı tanımlı hatalar bu kod ile başlar

Ayrıca Bakınız

Alım-Satım Sunucusunun Dönüş Kodları

Giriş ve Çıkış Sabitleri

Sabitler:

- Dosya açma bayrakları
- Dosya özellikleri
- Bir dosya içinde konumlandırma
- Kod sayfası kullanımı
- MessageBox

Dosya Açma Bayrakları

Dosya açma bayrakları, dosya erişim modunu belirler. Bayraklar şu şekilde tanımlanır:

Tanımlayıcı	Değer	Açıklama
FILE_READ	1	Dosya okuma amaçlı açılır. Bu bayrak FileOpen() fonksiyonunda kullanılır. Bir dosya açarken, FILE_WRITE ve/veya FILE_READ bayraklarının belirtilmesi gerekir.
FILE_WRITE	2	Dosya, yazma amacıyla açılır. Bu bayrak FileOpen() fonksiyonunda kullanılır. Bir dosya açarken, FILE_WRITE ve/veya FILE_READ bayraklarının belirtilmesi gerekir.
FILE_BIN	4	İkili oku/yaz modu (dizgiden dizgiye dönüşüm olmaksızın). Bu bayrak, FileOpen() fonksiyonunda kullanılır
FILE_CSV	8	CSV dosyası (tüm elemanları uygun tipte dizgilere dönüştürülür, unicode veya ansi ve bir ayraç ile ayrılır). Bu bayrak, FileOpen() fonksiyonunda kullanılır
FILE_TXT	16	Basit metin dosyası (csv dosyasına benzer ama ayraçlar yoktur). Bu bayrak, FileOpen() fonksiyonunda kullanılır
FILE_ANSI	32	ANSI tipli dizgiler (bir baytlık semboller). Bu bayrak, FileOpen() fonksiyonunda kullanılır
FILE_UNICODE	64	UNICODE tipi dizgiler (iki baytlık semboller). Bu bayrak, FileOpen() fonksiyonunda kullanılır
FILE_SHARE_READ	128	Birkaç programdan okunma amacıyla paylaşımlı erişim. Bu bayrak, FileOpen() fonksiyonunda kullanılır ama dosya açılması sırasında yine de FILE_WRITE ve/veya FILE_READ bayraklarının belirtilmesi gerekir.
FILE_SHARE_WRITE	256	Birkaç programdan yazma amaçlı paylaşımlı erişim. Bu bayrak, FileOpen() fonksiyonunda kullanılır ama dosya açılması sırasında yine de FILE_WRITE ve/veya FILE_READ bayraklarının belirtilmesi gerekir.
FILE_REWRITE	512	FileCopy() ve FileMove()fonksiyonları kullanılarak dosyanın yeniden yazılması olasılığı. Dosya mevcut olmalı veya yazma amaçlı açılmış olmalıdır, aksi durumda dosya açılmaz.
FILE_COMMON	4096	Tüm müşteri terminalleri için olan genel klasörün dosya yolu \Terminal\Common\Files. Bu bayrak, FileOpen(), FileCopy(), FileMove() ve FileIsExist() fonksiyonlarında kullanılır.

Dosya açılırken bir veya daha fazla bayrak belirtilebilir. Bu, bir bayrak kombinasyonu şeklinde olabilir. Bayrak kombinasyonu, listelenmiş bayraklar arasında mantıksal VEYA (|) işlemi kullanılarak yazılır. Örneğin, CSV biçimindeki bir dosyayı aynı anda hem okumak hem de yazmak amaçlı açmak için, FILE_READ|FILE_WRITE|FILE_CSV kombinasyonunu belirlenir.

Örnek:

```
int filehandle=FileOpen(filename,FILE_READ|FILE_WRITE|FILE_CSV);
```

Okuma ve yazma bayraklarını belirtirken, bazı belirli nitelikler bulunmaktadır:

- Eğer FILE_READ belirtilmişse, mevcut bir dosyayı açma girişimi gerçekleştirilir. Dosya mevcut değilse, açma işlemi başarısız olur; yeni bir dosya oluşturulmaz.
- FILE_READ|FILE_WRITE – Dosya mevcut değilse, belirtilen isimde yeni bir dosya oluşturulur.
- FILE_WRITE – Dosya, sıfır boyutlu olarak yeniden oluşturulur.

Bir dosya açılırken FILE_WRITE ve/veya FILE_READ bayraklarının belirtilmesi gerekir.

Açık bir dosyanın okuma tipini tanımlayan bayraklar öncelik sahibidir. FILE_CSV, önceliği en yüksek bayraktır, sonrasında FILE_BIN yer alır, en düşük öncelikli olan ise FILE_TXT bayrağıdır. Böylece, aynı anda birden çok bayrak belirlendiğinde (FILE_TXT|FILE_CSV veya FILE_TXT|FILE_BIN veya FILE_BIN|FILE_CSV), en yüksek öncelikli olan kullanılır.

Kodlama tipini belirten bayraklar da öncelik sahibidir. FILE_UNICODE, FILE_ANSI'den daha yüksek önceliğe sahiptir. Bu yüzden FILE_UNICODE|FILE_ANSI kombinasyonunu belirlerseniz, FILE_UNICODE bayrağı kullanılır.

FILE_UNICODE veya FILE_ANSI bayraklarından hiçbiri belirtilmemişse, FILE_UNICODE uygulanacaktır. FILE_CSV veya FILE_BIN veya FILE_TXT bayraklarından hiçbiri belirtilmemişse, FILE_CSV uygulanır.

Eğer bir dosya, metin dosyası şeklinde okuma amacıyla açılmışsa (FILE_TXT veya FILE_CSV) ve dosya başında 0xff,0xfe şeklinde iki baytlık özel bir gösterim bulunuyorsa; kodlama bayrağı FILE_ANSI olarak belirtilmiş olsa bile FILE_UNICODE kullanılır.

Ayrıca Bakınız

Dosya Fonksiyonları

Dosya Özellikleri

FileGetInteger() dosya özelliklerini elde etmek amacıyla kullanılır. İstenen özelliğin ENUM_FILE_PROPERTY_INTEGER sayımındaki tanımlayıcısı çağrı sırasında fonksiyona geçirilir.

ENUM_FILE_PROPERTY_INTEGER

Tanıtıcı	Tanımlayıcının açıklaması
FILE_EXISTS	Mevcudiyeti kontrol et
FILE_CREATE_DATE	Oluşturulma tarihi
FILE_MODIFY_DATE	En son değiştirilme tarihi
FILE_ACCESS_DATE	En son erişim tarihi
FILE_SIZE	Bayt bazında dosya boyutu
FILE_POSITION	İşaretçinin dosya içindeki konumu
FILE_END	Dosya sonu işaretini al
FILE_LINE_END	Satır sonu işaretini al
FILE_IS_COMMON	Dosya, tüm terminaller tarafından paylaşılan bir klasörde açılmış (bakınız FILE_COMMON)
FILE_IS_TEXT	Dosya, metin dosyası olarak açılmış (bakınız FILE_TXT)
FILE_IS_BINARY	Dosya, ikili dosya olarak açılmış (bakınız, FILE_BIN)
FILE_IS_CSV	Dosya, CSV dosyası olarak açılmış (bakınız FILE_CSV)
FILE_IS_ANSI	Dosya, ANSI şeklinde açılmış (bakınız FILE_ANSI)
FILE_IS_READABLE	Açılan dosya okunabilir özelliktedir (bakınız FILE_READ)
FILE_IS_WRITABLE	Açılan dosya yazılabilir özelliktedir (bakınız FILE_WRITE)

FileGetInteger() fonksiyonu için iki çağrı seçeneği bulunmaktadır. İlk seçenekte, dosya özeliklerini almak için, FileOpen() fonksiyonu ile dosya açılırken elde edilen tanıtıcı değer belirtilir. Bu seçenek, bir dosyanın tüm özelliklerinin elde edilmesini sağlar.

FileGetInteger() fonksiyonunda kullanılan ikinci seçenek, özellik değerine dosya ismi ile dönüş yapar. Bu seçenekte, sadece şu genel özellikler elde edilebilir:

- FILE_EXISTS - belirli bir isimdeki dosyanın mevcudiyeti
- FILE_CREATE_DATE - belirli bir isimdeki dosyanın oluşturulma tarihi
- FILE_MODIFY_DATE - belirli bir isimdeki dosyanın son değiştirilme tarihi
- FILE_ACCESS_DATE - belirli bir isimdeki dosyaya gerçekleştirilen son erişimin tarihi
- FILE_SIZE - belirli bir isimdeki dosyanın boyutu

Yukarıda belirtilenler dışındaki özelliklerin elde edilmeye çalışılması durumunda, FileGetInteger() çağrısı bir hata dönüşü gerçekleştirir.

Bir Dosyanın İçinde Konumlandırma

Çoğu dosya fonksiyonu, veri okuma/yazma işlemleriyle ilişkilidir. Aynı zamanda, FileSeek() fonksiyonunu kullanarak, bir dosya işaretçisinin konumunu, bir sonraki yazma veya okuma işleminin gerçekleştirileceği dosya içindeki bir konum olarak belirleyebilirsiniz. ENUM_FILE_POSITION sayımı, bir sonraki işlem için bayt bazında kaydırma işlemiyle belirleyebileceğiniz geçerli işaretçi pozisyonunu içerir.

ENUM_FILE_POSITION

Tanımlayıcı	Açıklama
SEEK_SET	Dosya başlangıcı
SEEK_CUR	Dosya işaretçisinin mevcut konumu
SEEK_END	Dosya sonu

Ayrıca Bakınız

FileIsEnding, FileIsLineEnding

Dizgi Dönüşüm İşlemlerinde Kod Sayfasının Kullanımı

string tipi değişkenleri char tipi dizilere dönüştürülürken veya tersi için, Windows işletim sistemindeki ANSI karakter setine karşılık gelen varsayılan kodlama (CP_ACP), MQL5 dilinde kullanılır. Eğer farklı bir kodlama tipi ayarlamak istiyorsanız, bunu CharArrayToString(), StringToCharArray() ve FileOpen() fonksiyonlarında bir ek parametre şeklinde belirtebilirsiniz.

Aşağıdaki tabloda, en popüler kod sayfaları için mevcut yerleşik sabitler listelenmiştir. Liste dışında kalan kod sayfaları, karşılık gelen bir kod ile belirtilebilir.

Yerleşik Kod Sayfası Sabitleri

Sabit	Değer	Açıklama
CP_ACP	0	Mevcut Windows ANSI kod sayfası.
CP_OEMCP	1	Mevcut sistemin OEM kod sayfası.
CP_MACCP	2	Mevcut sistemin Macintosh kod sayfası. **Not:** Bu değer, sadece eskiden yazılmış programlarda sıklıkla kullanılır ve artık gerekli değildir; bir süredir Macintosh bilgisayarlarda Unicode kullanılmaktadır
CP_THREAD_ACP	3	Geçerli iş parçacığı için Windows ANSI kod sayfası.
CP_SYMBOL	42	Sembol kod sayfası
CP_UTF7	65000	UTF-7 kod sayfası.
CP_UTF8	65001	UTF-8 kod sayfası.

Ayrıca Bakınız

Müşteri Terminali Özellikleri

MessageBox Diyalog Penceresinin Sabitleri

Bu bölüm, MessageBox() fonksiyonlarının dönüş kodlarını içermektedir. Mesaj penceresi bir İptal Et (Cancel) düğmesine sahipse, bu düğmeye veya ESC tuşuna basılması durumunda fonksiyon, IDCANCEL dönüşü yapar. Eğer mesaj penceresinde bir İptal Et düğmesine yoksa, ESC tuşuna basılmasının bir etkisi olmayacaktır.

Sabit	Değer	Açıklama
IDOK	1	"Tamam" (OK) düğmesine basılmış
IDCANCEL	2	"İptal Et" (Cancel) düğmesine basılmış
IDABORT	3	"Sonlandır" (Abort) düğmesine basılmış
IDRETRY	4	"Yeniden Dene" (Retry) düğmesine basılmış
IDIGNORE	5	"Gözardı Et" (Ignore) düğmesine basılmış
IDYES	6	"Evet" (Yes) düğmesine basılmış
IDNO	7	"Hayır" (No) düğmesine basılmış
IDTRYAGAIN	10	"Tekrar Dene" (Try Again) düğmesine basılmış
IDCONTINUE	11	"Devam Et" (Continue) düğmesine basılmış

MessageBox() fonksiyonunun ana bayrakları, diyalog penceresinin içeriğini ve davranışlarını tanımlar. Bu değer, aşağıdaki bayrak gruplarının bir kombinasyonu şeklinde olabilir:

Sabit	Değer	Açıklama
MB_OK	0x00000000	Sadece bir düğme içeren mesaj penceresi: OK. Default
MB_OKCANCEL	0x00000001	İki düğme içeren mesaj penceresi: OK ve Cancel
MB_ABORTRETRYIGNORE	0x00000002	Üç düğme içeren mesaj penceresi: Abort, Retry ve Ignore
MB_YESNOCANCEL	0x00000003	Üç düğme içeren mesaj penceresi: Yes, No ve Cancel
MB_YESNO	0x00000004	İki düğme içeren mesaj penceresi: Yes ve No
MB_RETRYCANCEL	0x00000005	İki düğme içeren mesaj penceresi: Retry ve Cancel
MB_CANCELTRYCONTINUE	0x00000006	Üç düğme içeren mesaj penceresi: Cancel, Try Again, Continue

Mesaj penceresinde bir ikon görüntülemek için, ek bayraklar belirtilmelidir:

Sabit	Değer	Açıklama
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	0x00000010	STOP işareti ikonu
MB_ICONQUESTION	0x00000020	Soru işareti ikonu
MB_ICONEXCLAMATION, MB_ICONWARNING	0x00000030	Ünlem işareti ikonu
MB_ICONINFORMATION, MB_ICONASTERISK	0x00000040	Daire içine alınmış i işareti

Ön tanımlı düğmeler, aşağıdaki bayraklarla tanımlanır:

Sabit	Değer	Açıklama
MB_DEFBUTTON1	0x00000000	İlk düğme MB_DEFBUTTON1 - eğer MB_DEFBUTTON2, MB_DEFBUTTON3 veya MB_DEFBUTTON4 belirtilmemişse ön tanımlıdır
MB_DEFBUTTON2	0x00000100	İkinci düğme ön tanımlıdır
MB_DEFBUTTON3	0x00000200	Üçüncü düğme ön tanımlıdır
MB_DEFBUTTON4	0x00000300	Dördüncü düğme ön tanımlıdır

MQL5 programları

Bir MQL5 programının çalıştırılabilmesi için öncelikle programın derlenmesi gerekir (Derle düğmesi veya F7 tuşu ile). Derleme süreci hatasız sonuçlanmalıdır (Uyarılar çalışmayı engellemez; ama analiz edilmelidirler). Bu süreçte, aynı isme ve EX5 uzantısına sahip çalıştırılabilen bir dosya, karşılık gelen dizinde oluşturulmalıdır - terminal_dizini\MQL5\Experts, terminal_dizini\MQL5\indicators veya terminal_dizini\MQL5\scripts. Böyle bir dosya çalıştırılabilir.

MQL5 programlarının çalışma özellikleri aşağıda gösterilmiştir:

- Program çalışması – ön-tanımlı olay işleyicilerinin çağrı sırası.
- Alım-satım stratejilerinin sınanması – MQL5 programlarının Strateji Sınama aracındaki işletim özellikleri.
- Müşteri terminali olayları – programlarda işlenebilen olayların açıklaması.
- İçe-aktarılmış fonksiyonların çağrılması – içe-aktarılmış fonksiyonlar için, tarif sırası, izin verilen parametreler, arama detayları ve çağrı koşulları.
- Çalışma-zamanı hataları – kritik hatalar ve çalışma-zamanı hataları hakkında bilgi edinme.

Uzman Danışmanlar, özel göstergeler ve scriptler, Taşı-ve-Bırak yöntemi ile Klavuz penceresinden herhangi bir çizelgeye eklenebilirler.

Bir Uzman Danışmanın çalışmasını durdurmak için, çizelgeden kaldırılması gerekir. Bunun için çizelge içerik menüsündeki "Uzmanlar Listesini" açın ve kaldırmak istediğiniz Uzman Danışmanı seçip "Kaldır" düğmesine basın. Uzman danışmanın çalışma şekli ayrıca "AutoTrading" düğmesinin durumundan da etkilenir.

Bir özel göstergeyi durdurmak için de, benzer şekilde göstergenin çizelge üzerinden kaldırılması gerekir.

Özel göstergeler ve Uzman Danışmanlar, çizelge üzerinden açık bir şekilde kaldırılana kadar çalışmaya devam ederler; çizelgeye tutturulmuş Uzman Danışmanlar ve özel göstergeler hakkındaki bilgiler, müşteri terminali oturumlarının arasında kaydedilir.

Scriptler sadece bir defaya mahsus olarak çalıştırılır; işlemin bitmesinin hemen ardından veya çizelge durumunun değişmesiyle veya terminalin kapatılmasıyla sonlandırılır. Müşteri terminalinin yeniden başlatılmasının ardından scriptler çalıştırılmaz, çünkü bunlarla ilgili herhangi bir kayıt tutulmaz.

Bir çizelge üzerinde sadece bir Uzman Danışman ve sadece bir script çalıştırılabilir, bununla birlikte sınırsız sayıda özel gösterge çalıştırılabilir.

Hizmetlerin çalışması için bir grafiğe bağlı olması gerekmez; hizmetler, yardımcı fonksiyonları yerine getirmek üzere tasarlanmıştır. Örneğin, bir hizmette kullanıcı tanımlı bir sembol oluşturabilir, grafiğini açabilir, ağ fonksiyonlarını kullanarak sonsuz bir döngüde veri alabilir ve sürekli olarak güncelleyebilirsiniz.

Program Çalıştırma

Her bir komut dosyası, servis ve Uzman Danışman ayrı bir iş parçacığında çalışır. Bir sembolde hesaplanan tüm göstergeler, farklı grafiklere eklense bile, aynı iş parçacığında çalışır. Böylece, bir semboldeki tüm göstergeler bir iş parçacığının kaynaklarını paylaşır.

Tiklerin işlenmesi ve geçmiş senkronizasyonu gibi, bir sembolle ilişkili tüm diğer eylemler de tutarlı bir şekilde, göstergeler ile aynı iş parçacığında gerçekleştirilir. Yani gösterge içinde sonsuz bir eylemin gerçekleştirilmesi durumunda, sembolle ilişkili diğer olayların hiçbiri gerçekleştirilmez.

Bir Uzman Danışmanı çalıştırırken, gerçek bir alım-satım ortamına sahip olduğundan, istenen sembol ve periyodun geçmişine erişim gerçekleştirebildiğinden ve terminal ile sunucu arasındaki veriyi senkronize edebildiğinden emin olun. Uzman Danışmanın mevcut veri ile başlatılmasının ardından tüm bu prosedürler için, terminal 5 saniyeden uzun olmayan bir gecikme oluşturur. Bu yüzden, sunucu bağlantısının olmaması durumunda, Uzman Danışmanın başlangıcında gecikmeler yaşanabilir.

Aşağıdaki tablo MQL5 programlarının kısa bir açıklamasını içerir:

MQL5 program	Çalışma	Not
Hizmet	Ayrı bir iş parçacığı, hizmetler için iş parçacığı sayısı hizmet sayısına eşittir	Döngüsel bir servis diğer programların çalışmasını bozamaz
Script	Ayrı iş parçacığı; gereken iş parçacıklarının sayısı, script sayısına eşittir	Döngülü bir script, diğer programların çalışmasını etkilemez
Uzman Danışman	Ayrı iş parçacığı; gereken iş parçacıklarının sayısı, Uzman Danışman sayısına eşittir	Döngülü bir Uzman Danışman, diğer programların çalışmasını etkilemez
Gösterge	Bir sembol üzerindeki tüm göstergeler için tek bir iş parçacığı. Gereken iş parçacıklarının sayısı, göstergeli sembollerin sayısına eşittir	Gösterge içindeki sonsuz bir döngü, sembol üzerindeki tüm diğer göstergeleri durdurur

Bir program, çizelgeye eklenmesinin hemen ardından, global değişkenlerin başlatılmasıyla birlikte, müşteri terminalinin belleğine yüklenir. Sınıf tipli bir global değişkenin bir yapıcısı varsa, bu yapıcı global değişkenlerin başlatılmasıyla birlikte çağrılacaktır.

Bunun ardından program, müşteri terminalinden gelecek bir olayı beklemeye başlar. Her MQL5 programı en az bir olay işleyicisine sahip olmalıdır, aksi durumda yüklenen program çalıştırılmayacaktır. Olay işleyicileri ön-tanımlı isimlere, parametrelere ve dönüş tiplerine sahiptir.

Tip	Fonksiyon ismi	Parametreler	Uygulama	Yorum
int	OnInit	yok	Uzman Danışmanlar ve göstergeler	Init olay işleyicisi. void dönüş tipinin kullanımını destekler.

Tip	Fonksiyon ismi	Parametreler	Uygulama	Yorum
void	OnDeinit	const int reason	Uzman Danışmanlar ve göstergeler	Deinit olay işleyicisi.
void	OnStart	yok	komut dosyaları ve hizmetler	Start olay yöneticisi.
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[]	göstergeler	tüm fiyatlar için Calculate olay işleyicisi.
int	OnCalculate	const int rates_total, const int prev_calculated, const int begin, const double &price[]	göstergeler	Calculate olay işleyicisi - tek bir veri dizisi için. Göstergeler aynı anda iki olay işleyicisine sahip olamazlar. BU durumda, veri dizisi üzerinde sadece bir olay işleyicisi çalışacaktır.
void	OnTick	yok	Uzman Danışmanlar	NewTick olay işleyicisi. Yeni tik olayı işlendiği sırada bu türden başka bir olay kabul edilmez.
void	OnTimer	yok	Uzman Danışmanlar ve göstergeler	Timer olay işleyicisi.
void	OnTrade	yok	Uzman Danışmanlar	Trade olay işleyicisi.
double	OnTester	yok	Uzman Danışmanlar	Tester olay işleyicisi.
void	OnChartEvent	const int id, const long &lparam, const double &dparam, const string &sparam	Uzman Danışmanlar ve göstergeler	ChartEvent olay işleyicisi.

Tip	Fonksiyon ismi	Parametreler	Uygulama	Yorum
void	OnBookEvent	const string &symbol_name	Uzman Danışmanlar ve göstergeler	BookEvent olay işleyici.

Müşteri terminali karşılık gelen açık durumdaki çizelgeye yeni olaylar gönderir. Olaylar, çizelgeler tarafından (çizelge olayları) veya MQL5 programları tarafından (özel olaylar) da gönderilebilir. Bir çizelge üzerindeki grafiksel nesnelerin oluşturulması ve silinmesi olayları, CHART_EVENT_OBJECT_CREATE ve CHART_EVENT_OBJECT_DELETE çizelge özelliklerinin ayarlanmasıyla devreye sokulabilir veya devre dışı bırakılabilir. Her MQL5 programı ve her çizelge kendilerine ait (yeni gelen olayların eklendiği) olay kuyruklarına sahiptir.

Bir program, sadece üzerinde çalıştığı çizelgedeki olayları alır. Tüm olaylar, alım sıralarına göre, birbiri ardına işlenirler. Eğer bir kuyrukta NewTick olayı zaten bulunuyorsa veya bu olay işlenmekteyse, yeni NewTick olayı MQL5 programının olay kuyruğuna eklenmeyecektir. Aynı şekilde, eğer ChartEvent olayı zaten kuyrukta ise, veya bu olay işlenmekteyse, bu türde yeni bir olay kuyruğa eklenmeyecektir. Zamanlayıcı olayları da aynı şekilde işlenir - eğer kuyrukta bir Timer olayı varsa veya bu olay işlenmekteyse, yeni zamanlayıcı olayı kuyruğa eklenmez.

Olay kuyrukları sınırlı ama yeterli büyüklüğe sahiptir, bu yüzden kuyruk aşımı durumu nadiren gerçekleşir. Kuyruk aşımı durumunda, yeni olaylar gözardı edilir.

Olayları yönetmek için sonsuz döngüler kullanmamanız şiddetle önerilir. Muhtemel istisnalar, tek bir Start olayını yöneten komut dosyaları ve hizmetlerdir.

Kütüphaneler olayları işlemezler.

Göstergelerde ve Uzman Danışmanlarda İzin verilmeyen Fonksiyonlar

Göstergeler, scriptler ve Uzman Danışmanlar, MQL5 dilinde yazılan çalıştırılabilir programlardır. Farklı görev tipleri için tasarlanmışlardır. Bu yüzden, program tipine bağlı olarak, bazı fonksiyonların kullanımında sınırlamalar vardır. Aşağıdaki fonksiyonların göstergelerde kullanılmasına izin verilmez:

- OrderCalcMargin();
- OrderCalcProfit();
- OrderCheck();
- OrderSend();
- SendFTP();
- Sleep();
- ExpertRemove();
- MessageBox().

Göstergeler için tasarlanmış olan ve Uzman Danışman ve scriptlerde kullanımına izin verilmeyen fonksiyonlar:

- SetIndexBuffer();

- IndicatorSetDouble();
- IndicatorSetInteger();
- IndicatorSetString();
- PlotIndexSetDouble();
- PlotIndexSetInteger();
- PlotIndexSetString();
- PlotIndexGetInteger.

Kütüphaneler bağımsız programlar değildir ve onları çağıran başka bir MQL5 programı (scriptler, Uzman Danışmanlar, vaya göstergeler) içinde çalıştırılırlar. Bu nedenle, yukarıdaki kısıtlamalar çağrılan kütüphaneye de uygulanır.

Hizmetlerde yasaklanan fonksiyonlar

Hizmetler, bir grafiğe bağlı olmadıkları için hiçbir olayı kabul etmemektedir. Hizmetlerde aşağıdaki fonksiyonlar yasaktır:

ExpertRemove();

EventSetMillisecondTimer();

EventSetTimer();

EventKillTimer();

SetIndexBuffer();

IndicatorSetDouble();

IndicatorSetInteger();

IndicatorSetString();

PlotIndexSetDouble();

PlotIndexSetInteger();

PlotIndexSetString();

PlotIndexGetInteger();

Göstergelerin Yüklenmesi ve Kaldırılması

Göstergeler şu durumlarda yüklenirler:

- bir göstergenin çizelgeye eklenmesi;
- terminal başlangıcı (gösterge terminal kapanmadan önce çizelgeye eklenmişse);
- bir şablonun yüklenmesi (göstergenin eklendiği çizelge, şablonda belirtilmişse);
- profilin değişimi (gösterge profil çizelgelerinden birine eklenmişse);
- göstergenin eklendiği çizelgenin sembolünün veya zaman aralığının değiştirilmesi;
- terminalin bağlı olduğu hesabın değiştirilmesi;
- göstergenin başarılı bir şekilde yeniden derlenmesi sonucunda, eğer gösterge çizelgeye eklenmiş durumdaysa;

- göstergenin giriş parametrelerinin değiştirilmesi.

Göstergeler şu durumlarda çizelgeden kaldırılırlar:

- göstegenin çizelgeden açık yolla kaldırılması;
- terminalin kapatılması (gösterge çizelgeye ekli durumdaysa);
- bir şablonun yüklenmesi durumunda, gösterge çizelgeye ekli durumdaysa;
- göstergenin eklendiği çizelgenin kapatılması;
- profil değişimi, gösterge değiştirilen profildeki çizelgeye ekli durumdaysa;
- göstergenin eklendiği çizelgenin sembolünün veya zaman aralığının değiştirilmesi;
- terminalin bağlı olduğu hesabın değiştirilmesi;
- göstergenin giriş parametrelerinin değiştirilmesi.

Uzman Danışmanların Yüklenmesi ve Kaldırılması

Uzman danışmanlar şu durumlarda yüklenirler:

- çizelgeye bir Uzman Danışman eklendiğinde;
- terminalin başlatılması (Uzman Danışman terminal kapanmadan önce çizelgeye eklenmişse);
- bir şablonun yüklenmesi (Uzman Danışmanın eklendiği çizelge, şablonda belirtilmişse);
- profil değişimi (Uzman Danışman profil çizelgelerinden birine eklenmişse);
- bir hesaba bağlanılması, hesap numarası aynı olsa bile (Uzman Danışman, terminal sunucu üzerinde yetkilendirilmeden öce çizelgeye eklenmişse).

Uzman Danışmanlar şu durumlarda kaldırılırlar:

- Uzman Danışmanın açık şekilde çizelgeden kaldırılması;
- eğer çizelge üzerinde zaten bir Uzman Danışman mevcutsa, yeni bir Uzman Danışmanın çizelgeye eklenmesiyle eskisi kaldırılacaktır.
- terminalin kapatılması (eğer Uzman Danışman çizelgeye ekli durumdaysa);
- bir şablonun yüklenmesi (eğer Uzman Danışman çizelgeye ekli durumdaysa);
- Uzman Danışmanın eklendiği çizelgenin kapatılması.
- profil değişimi; Uzman Danışman değiştirilen profildeki çizelgeye ekli durumdaysa; profile;
- terminalin bağlı olduğu hesabın değiştirilmesi (Uzman Danışman, terminal sunucu üzerinde yetkilendirilmeden öce çizelgeye eklenmişse);
- calling the ExpertRemove() function.

Uzman Danışmanın eklendiği çizelgenin zaman aralığı veya sembolünün değişmesi durumunda Uzman Danışman yüklenmez veya kaldırılmaz. bu durumda müşteri terminali sırayla eski sembol/zaman aralığı için OnDeinit() işleyicisini ve sembol/zaman aralığı için (eğer varsa) OnInit() işleyicisini çağırır; global değişkenlerin ve statik değişkenlerin değerleri sıfırlanmaz. Uzman Danışmanın başlatılmasından önceki tüm olaylar (OnInit() fonksiyonu) atlanır.

Scriptlerin Yüklenmesi ve Kaldırılması

Scriptler bir çizelgeye eklenmelerinin hemen ardından yüklenirler ve işlemlerinin bitmesiyle kaldırılırlar. OnInit() ve OnDeinit() fonksiyonları scriptlerde çağrılmaz.

Bir program kaldırıldığında (çizelge üzerinden silindiğinde), müşteri terminali global değişkenleri sonlandırır ve olay kuyruğunu siler. Bu durumda, sonlandırma, string tipli değişkenlerin sıfırlanması, dinamik dizi nesnelerinin bellek alanlarının boşaltılması ve değişken iseler yıkıcılarının çağrılması anlamına gelir.

Hizmetleri Yükleme ve Kaldırma

Eğer hizmetler terminal kapatılırken başlatıldılarsa, bu hizmetler terminal başlatıldıktan hemen sonra yüklenir. Hizmetler işlerini tamamladıktan hemen sonra kaldırılır.

Hizmetlerin, örneğin ağ fonksiyonlarını kullanarak kullanıcı tanımlı semboller oluşturma ve güncelleme gibi, sonsuz bir veri alma ve işleme döngüsü uygulayabileceğiniz tek bir OnStart() işleyicisi vardır.

Uzman Danışmanlar, göstergeler ve komut dosyalarından farklı olarak, hizmetler belirli bir grafiğe bağlı değildir, bu nedenle hizmetleri başlatmak için ayrı bir mekanizma sağlanmaktadır. Kılavuz'da "Hizmet Ekle" komutu kullanılarak yeni bir hizmet örneği oluşturulur. Bir hizmet örneği, uygun örnek menüsü kullanılarak başlatılabilir, durdurulabilir ve kaldırılabilir. Tüm örnekleri yönetmek için hizmet menüsünü kullanın.

Uzman Danışman İşleminin daha iyi anlaşılması için aşağıdaki kodun derlenmesi ve yükleme/kaldırma, şablon değişimi, sembol değişimi, zaman aralığının değişimi vb. işlemlerinin gerçekleştirilmesi tavsiye edilir

Örnek:

```
//+------------------------------------------------------------------+
//|                                                   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;
//+------------------------------------------------------------------+
```

```
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   Print("Başlatma");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print("Sonlandırma sebebi",reason);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

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

Ayrıca Bakınız

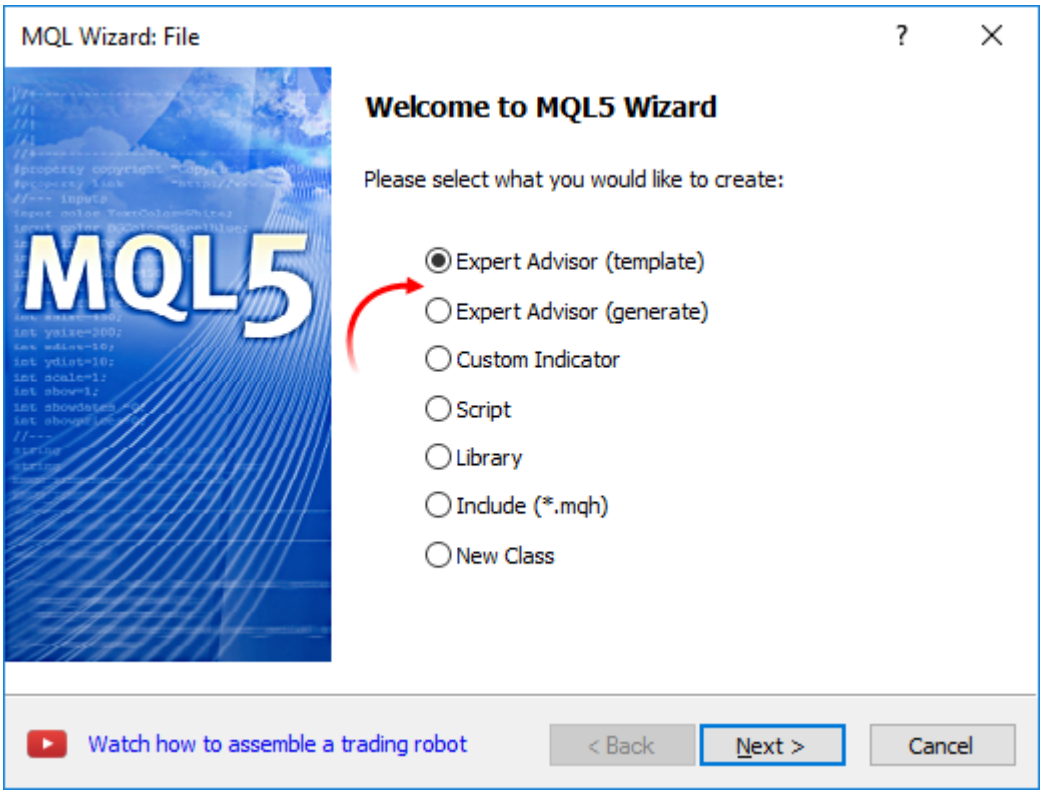
Müşteri terminali olayları, Olay işleyiciler

Alım-Satım İzni

Alım-Satım Otomasyonu

MQL5 dili, otomatik alım-satım sistemleri geliştirmek için tasarlanmış özel bir alım-satım fonksiyonları grubu içerir. İnsan müdahalesi olmadan otomatik alım-satım yapmak için geliştirilen programlara Uzman Danışman veya alım-satım robotu denir. MetaEditor ile bir Uzman Danışman oluşturabilmek için MQL5 Sihirbazını çalıştırın ve şu iki seçenekten birini seçin:

- Uzman Danışman (şablon) - kullanıma hazır olay işleme fonksiyonları ile programlamanın tüm gereksinimlerini karşılayabilecek bir şablon oluşturmanızı sağlar.
- Uzman Danışman (oluştur) - eksiksiz bir alım-satım robotu geliştirmek için basitçe gerekli modulleri seçmeniz yeterlidir: alım-satım sinyal modülleri, para yönetimi modülü ve İz-süren Stop modülü.



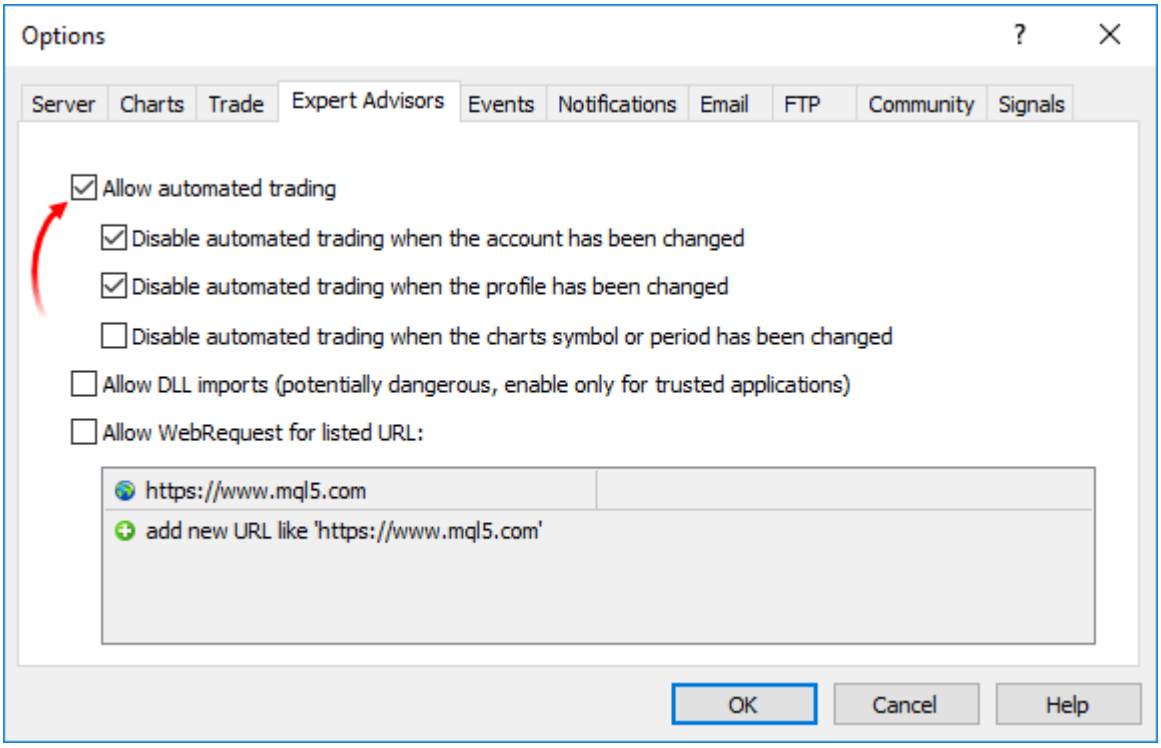
Alım-satım fonksiyonları sadece Uman Danışmanların ve betiklerin içinde kullanılblir. Göstergelerin alım-satım yapmasına izin verilmez.

Otomatik Alım-Satım için İşlem İzninin Denetlenmesi

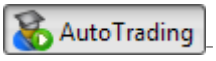
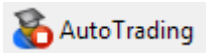
İnsan müdahalesi olmadan çalışabilen güvenilir bir Uzman Danışman geliştirmek için bazı önemli denetimlerin yapılması gerekir. Öncelikle, alım-stıma izin verilip verilmediğini program kapsamında denetlemeliyiz. Bu, otomatik sistemleri geliştirirken kullanılan en basit ve zorunlu denetimdir.

Terminalde otomatik alım-satım yapmak için işlem izninin denetlenmesi

Terminal ayarları tüm programlar için otomatik alım-satım izni vermenizi veya bu izni kaldırmanızı sağlar.



Terminalin Standard panelini kullanarak otomatik alım-satım seçeneğini değiştirebilirsiniz:

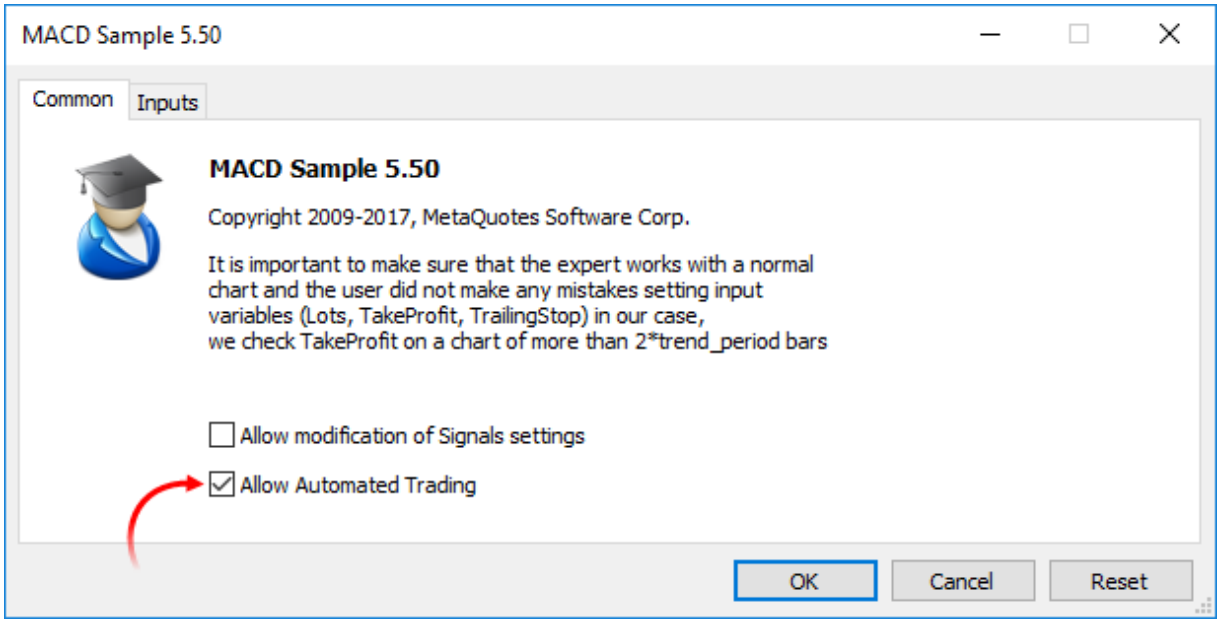
-  - otomatik alım-satım etkin, çalıştırılan programlarda alım-satım fonksiyonları kullanılabilir.
-  - otomatik alım-satım devre dışı, çalıştırılan programlarda alım-satım fonksiyonları kullanılamaz.

Denetim örneği

```
if (!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
   Alert("Terminal ayarlarında alım-satım izni etkin mi kontrol et!");
```

Belli bir Uzman Danışman/betik için alım-satım izninin denetimi

Bir programı çalıştırdığınızda o programa özel olarak alım-satımı yasaklayabilirsiniz. Bunun için program özelliklerindeki onay kutusunu kullanın.



Denetim örneği

```
if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
   Alert("Terminal ayarlarında alım-satım izni etkin mi kontrol et!");
else
  {
   if(!MQLInfoInteger(MQL_TRADE_ALLOWED))
      Alert("Otomatik alım-satım işlemleri program özellikleri kullanılarak yasakla
  }
```

Mevcut hesap üzerindeki herhangi bir Uzman Danışman/betik için alım-satım izninin denetimi

Otomatik alım-satım işlemleri sunucu üzerinden de devre-dışı bırakılabilir. Denetim örneği

```
if(!AccountInfoInteger(ACCOUNT_TRADE_EXPERT))
   Alert(,AccountInfoInteger(ACCOUNT_LOGIN),
   " hesabı için otomatik alım-satım izni sunucu üzerinde devre-dışı bırakılmış");
```

Otomatik alım-satım işlemleri hesap genelinde devredışı bırakılmışsa Uzman Danışmanlardaki/betiklerdeki alım-satım işlemleri çalıştırılmaz.

Mevcut hesap üzerinde alım-satım izninin denetimi

Bazı durumlarda alım-satım işlemleri belli hesaplar için devre-dışı bırakılır - alım-satım işlemleri elle veya otomatik olarak yapılamaz. Hesaba yatırımcı şifresi ile giriş yapılıp yapılmadığına dair şuna benzer denetim kullanılabilir:

```
if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
   Comment("hesap üzerinde alım-satım devre-dışı bırakılmış ",AccountInfoInteger(AC
         ".\n Hesaba yatırımcı şifresi ile giriş yapılmış olabilir.",
         "\n Terminal günlüğünü şu giriş için denetle:",
```

```
            "\n\'",AccountInfoInteger(ACCOUNT_LOGIN),"\': trading has been disabled -
```

AccountInfoInteger(ACCOUNT_TRADE_ALLOWED) çağrısı şu durumlarda **false** dönüşü yapabilir:

- alım-satım sunucusuyla bağlantı kurulamıyor. Bu durum TerminalInfoInteger(TERMINAL_CONNECTED) çağrısı ile kontrol edilebilir;
- alım-satım hesabı salt-okunur moda çevrilmiş (arşive gönderilmiş);
- hesap üzerinde alım-satım işlemleri sunucu üzerinden devre-dışı bırakılmış;
- hesaba yatırımcı modu ile bağlanılmış.

Ayrıca bakınız

Müşteri Terminalinin Özellikleri, Hesap Özellikleri, Çalışan bir MQL5 Programının Özellikleri

Müşteri Terminali Olayları

Init

Programların (bir Uzman Danışmanın veya özel göstergenin) müşteri terminaline yüklenmesinin hemen ardından global değişkenlerin başlatılma sürecine geçilir ve Init olayı, - eğer mevcutsa - OnInit() içinde işlenmesi için gönderilir. Bu olay, bir finansal enstrümanda ve/veya bir çizelgede değişiklik olduğunda, program MetaEditor'de yeniden derlendiğinde, giriş parametreleri Uzman Danışmanın veya göstergenin başlangıç penceresinden değiştirildiğinde de oluşturulur. Uzman Danışmanlar hesap değişimi yapıldığında da başlatılırlar. Init olayı scriptler için oluşturulmaz.

Deinit

Global değişkenlerin sonlandırılmasından ve programın (Uzman Danışmanın veya özel göstergenin) kaldırılmasından sonra, müşteri terminali Deinit olayını programa gönderir. Deinit olayı ayrıca, müşteri terminali veya bir çizelge kapatıldığında, sembol ve/veya zaman aralığı değiştirilmeden önce, programın başarıyla yeniden derlenmesi durumunda, giriş parametrelerinin değiştirilmesi durumunda ve hesap değiştirildiğinde de oluşturulur.

Sonlandırma sebebi, OnDeinit() fonksiyonuna geçirilen bir parametre ile alınabilir. OnDeinit() fonksiyonunun çalışma süresi 2.5 saniye ile sınırlandırılmıştır. Fonksiyon bu süre içinde tamamlanmazsa, zorla sonlandırılır. Deinit olayı, scriptlerde oluşturulmaz.

Start

Start, bir komut dosyasını veya hizmeti yükledikten sonra başlatmak için özel bir olaydır. OnStart fonksiyonu tarafından yönetilir. Start olayı Uzman Danışmanlara ve özel göstergelere iletilmez.

NewTick

NewTick olayı, yeni fiyat teklifleri geldiğinde oluşturulur, Uzman Danışmanlara yerleştirilen OnTick() fonksiyonu ile işlenir. OnTick fonksiyonu bir önceki fiyatı işlerken yeni bir fiyatın alınması durumunda, yeni fiyat, Uzman Danışman tarafından göz ardı edilir; söz konusu olay için bir kuyruk oluşturulmaz.

OnTick() fonksiyonunun işlemi tamamlanana kadar tüm yeni fiyatlar gözardı edilir. Sonrasında ise, fonksiyon sadece yeni bir fiyat gelirse çalışır. NewTick olayı, otomatik alım-satım iznine bakılmaksızın oluşturulur ("Otomatik alım-satıma izin ver/engelle" düğmesi). Otomatik alım-satımın engellenmesi, Uzman Danışmanın çalışması sırasında alım-satım isteği göndermesine izin verilmediği anlamına gelir.

Uygun düğmeye basılarak Otomatik alım-satımın engellenmesi, OnTick() fonksiyonunun mevcut çalışmasını durdurmaz.

Calculate

Calculate olayı, Init olayının hemen ardından fiyat verisinde herhangi bir değişik olduğunda oluşturulur. OnCalculate fonksiyonu ile işlenir.

Timer

Timer olayı, EventSetTimer fonksiyonu ile zamanlayıcıyı devreye sokan Uzman danışman için, müşteri terminali tarafından periyodik olarak oluşturulur. Bu fonksiyon genellikle OnInit içerisinde çağrılır. Timer olayı, OnTimer fonksiyonu ile işlenir. Uzman Danışman işleminin tamamlanmasının ardından,

zamanlayıcının EventKillTimer fonksiyonu ile yok edilmesi gerekir, bu fonksiyon genellikle OnDeinit fonksiyonu içerisinde çağrılır.

Trade

Trade olayı, bir alım-satım işleminin sunucu üzerinde tamamlanmasıyla oluşturulur. Trade olayı OnTrade() fonksiyonu ile, şu alım-satım işlemleri için işlenir:

- bir bekleyen emrin gönderilmesi, değiştirilmesi veya silinmesi;
- zaman-aşımı veya yetersiz bakiye nedeniyle bir bekleyen emrin sonlandırılması;
- Bir bekleyen emrin aktif hale getirilmesi;
- bir pozisyonun (tamamının veya bir kısmının) açılması, eklenmesi veya kapatılması;
- Açık bir pozisyonun değiştirilmesi (durdurma noktalarının değiştirilmesi - Stop Loss ve/veya Take Profit).

TradeTransaction

Alım-satım hesabı üzerinde bazı kesin işlemler gerçekleştirildiği zaman hesabın durumu değişir. Bu eylemler şunları kapsamaktadır:

- OrderSend ve OrderSendAsync fonksiyonları kullanılarak herhangi bir MQL5 uygulamasından yapılan alım-satım istekleri ve bunların ilerideki kullanımları;
- Terminalin grafiksel arayüzü ile yapılan alım satım istekleri ve bunların ilerideki kullanımları;
- Sunucu üzerindeki bekleyen emir ve durdurma emri aktivasyonu;
- İşlemlerin alım-satım sunucusu tarafından gerçekleştirilmesi.

Yukarıda sayılanlar eylemlerin sonucunda şu alım-satım faaliyetleri gerçekleşir:

- alım-satım isteğinin işlenmesi;
- açık emirlerin değişimi;
- emir geçmişinin değişimi;
- işlem geçmişinin değişimi;
- pozisyonların değişimi.

Örneğin, bir alım emri gönderildiğinde, önce işlenir, hesap için uygun bir alım emri oluşturulur, gerçekleştirilir ve açık emir listesinden çıkarılır, sonra emir geçmişine eklenir, uygun bir işlem geçmişe eklenir ve yeni pozisyon açılır. Tüm bu eylemler alım-satım faaliyetidir. Böyle bir faaliyetin terminale ulaşımı ise TradeTransaction olayıdır. Bu olay OnTradeTransaction fonksiyonu ile işlenir.

Tester

Tester olayı, Uzman Danışmanın tarihsel veriyle sınanmasının ardından oluşturulur. Olay OnTester() fonksiyonu ile işlenir.

TesterInit

TesterInit olayı, strateji sınayıcıda optimizasyonun başlamasıyla, ilk optimizasyon geçişinden hemen önce oluşturulur. TesterInit olayı OnTesterInit() fonksiyonu ile işlenir.

TesterPass

TesterPass olayı, yeni bir veri çerçevesi alındığında oluşur. TesterPass olayı OnTesterPass() fonksiyonu ile işlenir.

TesterDeinit

The TesterDeinit olayı, strateji sınayıcıda Uzman Danışman optimizasyonunun bitmesinin ardından oluşturulur. TesterDeinit olayı, OnTesterDeinit() fonksiyonu ile işlenir.

ChartEvent

ChartEvent olayı müşteri terminali tarafından, kullanıcının çizelge üzerinde çalışması durumunda oluşturulur:

- klavye tuşuna basılması, çizelge penceresinin yakınlaştırılması;
- grafiksel nesne oluşturulması
- grafiksel nesnenin silinmesi
- çizelgedeki bir grafiksel nesnenin tıklanması
- grafiksel nesnenin fare ile taşıması
- LabelEdit içindeki metin düzenleme işleminin bitmesi.

Ayrıca, EventChartCustom fonksiyonunu kullanarak herhangi bir MQL5 programından bir Uzman Danışmana gönderilebilecek, bir ChartEvent özel (kullanıcı) olayı da mevcuttur. Bu olay OnChartEvent fonksiyonu ile işlenir.

BookEvent

BookEvent olayı Piyasa Derinliğindeki bir değişimin ardından müşteri terminali tarafından oluşturulur; OnBookEvent fonksiyonu ile işlenir. Belirtilen sembol için BookEvent olayını oluşturmak amacıyla, sembol, MarketBookAdd fonksiyonu kullanılarak bu olaya abone edilmelidir.

Belirtilen sembolün BookEvent aboneliğini sonlandırmak için, MarketBookRelease fonksiyonunun çağrılması gerekir. BookEvent olayı yayın-tipi bir olaydır - yani sadece bir Uzman Danışmanın olaya abone edilmesi yeterlidir, OnBookEvent işleyicisine sahip tüm diğer Uzman Danışmanlar da bu yayını alacaktır. Bu nedenle, işleyiciye parametre olarak geçirilen sembol isminin analiz edilmesi gerekir.

Ayrıca Bakınız

Olay işleyiciler, Programın çalışması

Kaynaklar

Grafik ve ses dosyalarını MQL5 programında kullanma

MQL5 dilinde yazılan programlar ses ve grafik dosyalarıyla çalışmaya olanak sağlar:

- PlaySound() fonksiyonu bir ses dosyasını yürütür;
- ObjectCreate() fonksiyonu ise, OBJ_BITMAP ve OBJ_BITMAP_LABEL grafiksel nesnelerini kullanarak, kullanıcı arayüzlerinin oluşturulmasına olanak sağlar.

PlaySound()

PlaySound() fonksiyonunun çağrı örneği:

```
//+------------------------------------------------------------------+
//| Standart OrderSend() çağrısı yapar ve bir ses oynatır            |
//+------------------------------------------------------------------+
void OrderSendWithAudio(MqlTradeRequest  &request, MqlTradeResult &result)
  {
  //--- sunucuya bir istek gönder
   OrderSend(request,result);
   //--- eğer istek kabul edilirse, Ok.wav ses dosyasını yürüt
   if(result.retcode==TRADE_RETCODE_PLACED) PlaySound("Ok.wav");
   //--- başarısızlık durumunda timeout.wav dosyasından alarm sesi yürüt
   else PlaySound("timeout.wav");
  }
```

Bu örnek standart terminal paketine eklenmiş olan 'Ok.wav' ve 'timeout.wav' ses dosyalarının nasıl oynatıldıklarını göstermektedir. Bu dosyalar **terminal_dizini\Sounds** klasörüne yerleştirilmiştir. Burada **terminal_dizini**, MetaTrader 5 Müşteri Terminalinin başlatıldığı klasördür. Terminal dizininin konumu bir mql5 programı içinde şu şekilde öğrenilebilir:

```
//--- Terminal verisinin depolandığı klasör
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
```

Sadece **terminal_dizini\Sounds**, klasöründeki ses dosyalarını değil aynı zamanda alt klasörlerdeki dosyalarıda kullanabilirsiniz örn: **terminal_veri_dizini\MQL5.** Terminal veri dizininin konumunu, terminal menüsünden "Dosya" -> "Veri Klasörünü Aç" veya şu komutunu kullanarak öğrenebilirsiniz:

```
//--- Terminal verisinin depolandığı klasör
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
```

Örneğin, Demo.wav ses dosyası terminal_veri_dizini\MQL5\Files konumunda ise, PlaySound() çağrısı şu şekilde yazılmalıdır:

```
//--- terminal_veri_dizini\MQL5\Files\Demo.wav konumundaki Demo.wav dosyasını oynat
   PlaySound("\\Files\\Demo.wav");
```

Yorum kısmında dosya adresinin "\" işaretiyle fonksiyon içinde ise "\\" şeklinde kullanıldığını lütfen not edin.

Dosya adresini belirtirken her zaman çift ters-bölü işareti kullanın. Tekli ters-bölü işareti, derleyicinin kaynak kodu içerisindeki sabit dizelerle ve karakter sabitleri ile çalışırken kullandığı bir kontrol elemanıdır.

Kayıttan yürütmeyi durdurmak için NULL parametresi ile PlaySound() fonksiyonunun çağrılması:

```
//--- PlaySound() fonksiyonunun NULL parametresi ile çağrılması kayıttan yürütmeyi du
   PlaySound(NULL);
```

ObjectCreate()

ObjectCreate() fonksiyonunu kullanarak bir grafiksel etiket (OBJ_BITMAP_LABEL) oluşturan Uzman Danışman örneği.

```
string label_name="currency_label";         // OBJ_BITMAP_LABEL nesnesinin ismi
string euro      ="\\Images\\euro.bmp";     // dosya adresi terminal_veri_dizini\MQL5\
string dollar    ="\\Images\\dollar.bmp";   // dosya adresi terminal_veri_dizini\MQL5\
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- hala oluşturulmamışsa bir OBJ_BITMAP_LABEL düğmesi oluştur
   if(ObjectFind(0,label_name)<0)
     {
      //--- OBJ_BITMAP_LABEL nesnesini oluluştur
      bool created=ObjectCreate(0,label_name,OBJ_BITMAP_LABEL,0,0,0);
      if(created)
        {
         //--- düğmeyi çizelgenin sol üst köşesine yerleştir
         ObjectSetInteger(0,label_name,OBJPROP_CORNER,CORNER_RIGHT_UPPER);
         //--- şimdi nesne özelliklerini ayarla
         ObjectSetInteger(0,label_name,OBJPROP_XDISTANCE,100);
         ObjectSetInteger(0,label_name,OBJPROP_YDISTANCE,50);
         //--- son hata kodunu sıfırla
         ResetLastError();
         //--- düğmenin "basılı" durumunu göstermek için bir resim indir
         bool set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,0,euro);
         //--- sonucu test et
         if(!set)
           {
            PrintFormat("Resim %s dosyasından indirilemedi. Hata kodu %d",euro,GetLast
           }
         ResetLastError();
         //--- düğmenin "serbest" durumunu göstermek için bir resim indir
         set=ObjectSetString(0,label_name,OBJPROP_BMPFILE,1,dollar);

         if(!set)
```

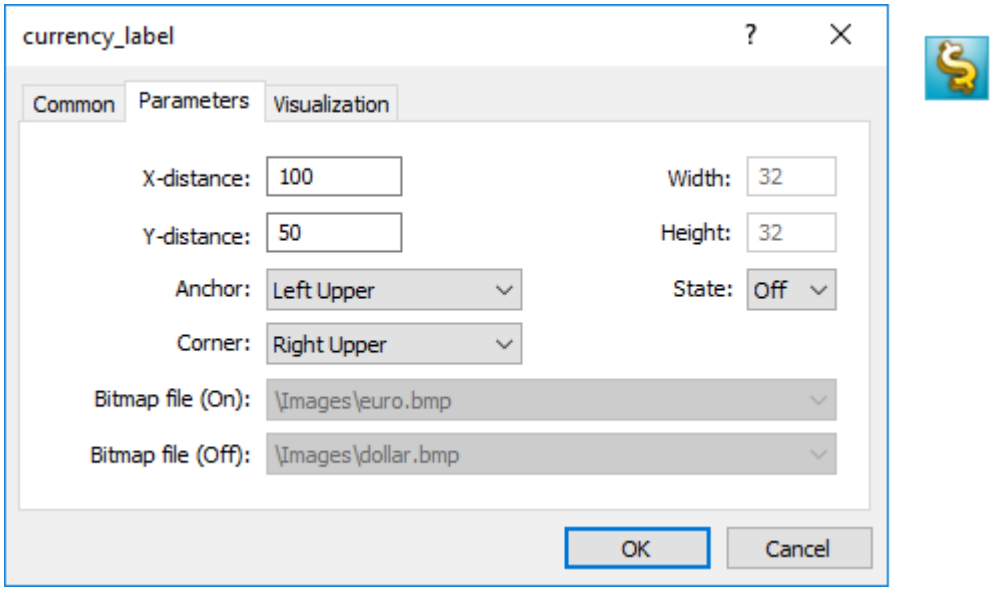
```
            {
             PrintFormat("Resim %s dosyasından indirilemedi. Hata kodu %d",dollar,GetLa
            }
          //--- düğmenin hemen gözükmesi için çizelgenin yenilenmesi komutunu gönder
          ChartRedraw(0);
         }
       else
         {
          //--- nesne oluşturulmadı, uyar
          PrintFormat("OBJ_BITMAP_LABEL nesnesi oluşturulamadı. Hata kodu %d",GetLastEr
         }
      }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- nesneyi çizelgeden sil
   ObjectDelete(0,label_name);
  }
```

currency_label nesnesinin oluşturulup başlatılması OnInit() fonksiyonu içinde gerçekleşir. Grafiksel dosyaların adresleri global değişkenler **euro** ve **dollar** içinde ayarlanır, bunun için çift ters bölü işareti kullanılır:

```
string euro      ="\\Images\\euro.bmp";    // dosyanın adresi terminal_veri_dizini\MQ
string dollar    ="\\Images\\dollar.bmp";  // dosyanın adresi terminal_veri_dizini\MQ
```

Dosyalar, **terminal_veri_dizini**\MQL5\Images klasörü içinde yer alırlar.

OBJ_BITMAP_LABEL nesnesi duruma (basılı veya serbest) bağlı olarak iki resimden birini gösteren bir düğmedir: euro.bmp veya dollar.bmp.



Grafiksel arayüze sahip nesnelerin boyutları, otomatik olarak resim boyutuna göre ayarlanır. OBJ_BITMAP_LABEL nesnesinin üzerine sol fare tuşu ile tıklandığında resim değiştirilir (**"Seçimi devre dışı bırak" özelliği belirtilmiş olmalıdır**). OBJ_BITMAP nesnesi de aynı şekilde oluşturulur - gereken resim ile birlikte arka-planı oluşturmak için kullanılır.

OBJ_BITMAP and OBJ_BITMAP_LABEL nesnelerinin görünümünden sorumlu olan OBJPROP_BMPFILE özelliği, dinamik olarak değiştirilebilir. Bu mql5 programları için çeşitli interaktif arayüzler oluşturulmasına imkan verir.

Mql5 programlarının derlenmesi sırasında kaynakların çalıştırılabilir dosyalara eklenmesi

Bir mql5 programı görüntü ve ses dosyaları şeklinde bir çok farklı yüklenebilir kaynağa ihtiyaç duyabilir. Çalıştırılabilir bir dosyanın MQL5 içine aktarılırken bu ihtiyacın giderilebilmesi için, #resource derleyici direktifi kullanılmalıdır:

```
#resource kaynak_dosyasının_adresi
```

#resource komutu, derleyiciye, **kaynak_dosya_adresi** ile belirtilen kaynağın çalıştırılabilir EX5 dosyasına eklenmesi gerektiğini söyler. Bu şekilde, gereken tüm ses ve görüntüler doğrudan EX5 dosyasına yerleştirilebilir ve programı farklı bir terminal üzerinde çalıştırmak istediğinizde söz konusu dosyaların ayrı ayrı aktarılmaları gerekmez. Her EX5 programı kaynak dosyalarını içerebilir ve yine her EX5 programı başka bir EX5 programındaki kaynakları kullanabilir.

BMP ve WAV dosyaları EX5 içine aktarılmadan önce otomatik olarak sıkıştırılırlar. Yani, MQL5 içinde programlar oluştururken ses ve görüntü kaynaklarının bu şekildeki kullanımı, MQL5 programının normal yolla yazılmasına nazaran, gerekli dosyaların toplam boyutunun daha iyi azaltılmasına olanak tanır.

Kaynak dosyasının boyutu 16 Mb'tan fazla olmamalıdır.

Belirtilen kaynağın derleyici ile aranması

Kaynaklar #resource "<kaynak dosyasının adresi>" komutu kullanılarak eklenirler

```
#resource "<kaynak_dosya_adresi>"
```

<kaynak_dosya_adresi> sabit dizesinin 63 karakteri geçmemesi gerekir.

Derleyici belli bir adresteki kaynağı şu sırayla arar:

- eğer, "\" ayracı ("\\" şeklinde yazılır) adresin başına yazılmışsa, arama **terminal_veri_dizini**\MQL5\, konumuna göre yapılır
- eğer ters-bölü işareti yoksa, kaynağı kaynak dosyasının yazıldığı konumda arar.

Kaynak adresi alt-dizeler "..\\" ve ":\\" içeremez

Kaynak ekleme örnekleri:

```
//--- kaynakların doğru şekilde belirtilmesi
#resource "\\Images\\euro.bmp" // euro.bmp, terminal_veri_dizini\MQL5\Images\ klasörü
#resource "picture.bmp"        // picture.bmp kaynak dosyasıyla aynı konumda
#resource "Resource\\map.bmp"  // kaynak, kaynak_dosa_dizini\Resource\map.bmp şeklind

//--- kaynakların yanlış belirtilmesi
#resource ":picture_2.bmp"     // ":"işaretini içermemeli
#resource "..\\picture_3.bmp"  // ".." içermemeli
#resource "\\Files\\Images\\Folder_First\\My_panel\\Labels\\too_long_path.bmp" // 63
```

Kaynakların Kullanımı

Kaynak ismi

Kaynağın bildirimi #resource direktifi ile yapıldıktan sonra programın her yerinde kullanılabilir. Kaynağın ismi konumunu belirleyen adrestir ve başında ters-bölü olmadan yazılır. Kod içerisinde kendi kaynağınızı kullanmak istiyorsanız "::" özel ismini kaynak isminden önce belirtmeniz gerekir.

Örnekler:

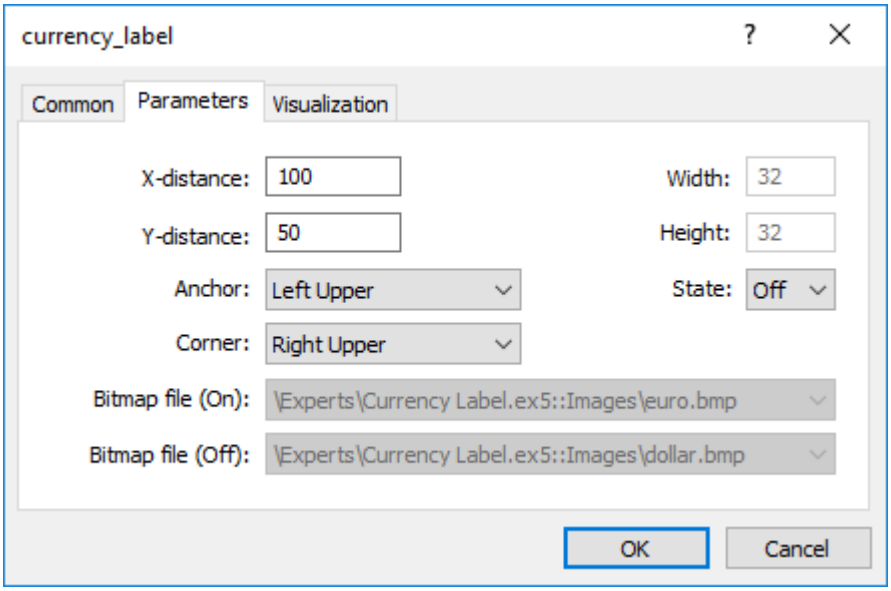
```
//--- kaynakların belirlenmesi ve isimlerinin yorum olarak verilmesi örneği
#resource "\\Images\\euro.bmp"          // kaynak ismi - Images\euro.bmp
#resource "picture.bmp"                 // kaynak ismi - picture.bmp
#resource "Resource\\map.bmp"           // kaynak ismi - Resource\map.bmp
#resource "\\Files\\Pictures\\good.bmp" // kaynak ismi - Files\Pictures\good.bmp
#resource "\\Files\\Demo.wav";          // kanak ismi - Files\Demo.wav"
#resource "\\Sounds\\thrill.wav";       // kaynak ismi - Sounds\thrill.wav"
...

//--- kaynakların kullanımı
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");
```

Bir kaynak ile OBJ_BITMAP ve OBJ_BITMAP_LABEL nesnelerine görüntü (resim) ayarlarken, OBJPROP_BMPFILE özelliğinin değerinin el yordamıyla değiştirilmesinin mümkün olmadığı not edilmelidir. Örneğin, OBJ_BITMAP_LABEL nesnesini oluşturmak için euro.bmp ve dollar.bmp kaynaklarını kullanırız.

```
#resource "\\Images\\euro.bmp";    // euro.bmp, terminal_veri_dizini\MQL5\Images\ kon
#resource "\\Images\\dollar.bmp";  // dollar.bmp, terminal_veri_dizini\MQL5\Images\ k
```

Bu nesnenin özeliklerini görüntülerken, BitMap Dosyası (On) ve BitMap Dosyası (Off) özelliklerinin soluklaştırıldığını ve el yordamıyla değiştirilemediğini görürüz:



Diğer mql5 programlarının kaynaklarının kullanılması

Kaynak kullanımının bir diğer avantajı daha vardır - her mql5 programında, diğer EX5 dosyalarının kaynakları kullanılabilir. Diğer bir deyişle, bir EX5 dosyasının kaynakları diğer bir çok mql5 programı içinde kullanılabilir.

Başka bir dosyadaki kaynağın ismini kullanmak için bunu <adres_EX5_dosya_ismi>::<kaynak_ismi> şeklinde belirtmelisiniz. Örnek olarak, Draw_Triangles_Script.mq5 betik dosyasının triangle.bmp dosyasındaki bir kaynağı içerdiğini düşünelim:

```
#resource "\\Files\\triangle.bmp"
```

Yani betik içinde kullanılacak isim "Files\triangle.bmp" şeklinde gözükecektir ve bunu kullanmak için "::" ifadesi kaynak ismine eklenmelidir.

```
//--- kaynağın betik içinde kullanımı
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"::Files\\triangle.bmp");
```

Aynı kaynağı başka bir programdan kullanmak istediğimizde (örneğin bir Uzman Danışmandan), EX5 dosyasının adresini **terminal_veri_dizini\MQL5** dizinine ve betiğin EX5 dosyasına göre (**Draw_Triangles_Script.ex5**) kaynak ismine eklememiz gerekir. Betiğin standart klasör (**terminal_veri_dizini\MQL5**Scripts\) içine yerleştirildiğini düşünelim, bu durumda çağrı şu şekilde yapılır:

```
//--- Bir betikteki kaynağın Uzman Danışman için kullanımı
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"\\Scripts\\Draw_Triangles_Script.e
```

Kaynak başka bir EX5 dosyasından alındığında, çalıştırılabilir dosyanın adresi belirtilmezse, çalıştırılabilir dosya kaynağın çağrıldığı dosya ile aynı konumda aranır. Yani, bir Uzman Danışman, Draw_Triangles_Script.ex5 betiğindeki bir kaynağı adres belirtmeden şu şekilde çağırıyorsa:

```
//--- Betik kaynağını Uzman Danışman içinde adres belirtmeden çağır
ObjectSetString(0,my_bitmap_name,OBJPROP_BMPFILE,0,"Draw_Triangles_Script.ex5::Files\\
```

Bu durumda, dosya **terminal_veri_dizini\MQL5**Experts\ klasöründe aranacaktır (eğer Uzman Danışman **terminal_veri_dizini\MQL5**Experts\ dizininde bulunuyorsa)

Kaynak şeklinde eklenen özel göstergelerle çalışma

MQL5 uygulamalarının çalışması için bir veya birkaç özel gösterge gerekebilir. Bunların kodları bir harici MQL5 programına eklenebilir. Göstergelerin kaynaklar şeklinde eklenmesi uygulamaların dağılımını basitleştirir.

Aşağıda, **terminal_veri_klasörü**\MQL5\Indicators\ dizinine yerleştirilmiş SampleIndicator.ex5 göstergesinin kaynak şeklinde eklenmesi örnek gösterilmiştir:

```
//+------------------------------------------------------------------+
//|                                                     SampleEA.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#resource "\\Indicators\\SampleIndicator.ex5"
int handle_ind;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   handle_ind=iCustom(_Symbol,_Period,"::Indicators\\SampleIndicator.ex5");
   if(handle_ind==INVALID_HANDLE)
     {
      Print("Uzman: iCustom çağrısı: Hata kodu=",GetLastError());
      return(INIT_FAILED);
     }
```

```
//--- ...
   return(INIT_SUCCEEDED);
  }
```

OnInit() fonksiyonunun içerisinde kendisinin kopyalarını oluşturan göstergeler özel ilgi gerektirir. Kaynağın şu şekilde belirtilmesi gerektiğini lütfen aklınızda tutun: **<veri_yolu_EX5_dosya_ismi>::<resource_name>**.

Örneğin, SampleIndicator.ex5 göstergesi SampleEA.ex5 uzmanına kaynak olarak eklenmişse, başlatma fonksiyonunun içinde iCustom() çağrısı yapılırken kullanılacak veri yolu şu şekilde olur: "\\Experts\\SampleEA.ex5::Indicators\\SampleIndicator.ex5". Veri yolu açıkca belirtildiğinde SampleIndicator.ex5 göstergesi SampleEA.ex5 uzmanına sıkı şekilde bağlanır ve Uzman Danışman ayrı çalışma yeteneğini yitirir.

Göstergenin veya uzmanın kendisine ait veri yolu GetRelativeProgramPath() fnksiyonu ile alınabilir. Aşağıdaki örnekte bu foksiyonun kullanımı gösterilmiştir:

```
//+------------------------------------------------------------------+
//|                                                SampleIndicator.mq5 |
//|                        Copyright 2013, MetaQuotes Software Corp. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property indicator_separate_window
#property indicator_plots 0
int handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- kendi veri yolunu almanın yanlış yolu
//--- string path="\\Experts\\SampleEA.ex5::Indicators\\SampleIndicator.ex5";
//--- kendi veri yolunu almanın doğru yolu
  string path=GetRelativeProgramPath();
//--- gösterge tamponlarının eşlenmesi
   handle=iCustom(_Symbol,_Period,path,0,0);
   if(handle==INVALID_HANDLE)
     {
      Print("Gösterge: iCustom çağrısı: Hata kodu=",GetLastError());
      return(INIT_FAILED);
     }
   else Print("Gösterge işleyicisi=",handle);
//---
   return(INIT_SUCCEEDED);
  }
///....
//+------------------------------------------------------------------+
//| Programın Göreli Veri Yolunu Al                                  |
//+------------------------------------------------------------------+
string GetRelativeProgramPath()
```

```
  {
   int pos2;
//--- uygulamanın kesin veri yolunu al
   string path=MQLInfoString(MQL_PROGRAM_PATH);
//--- "\MQL5\" alt dizgisinin pozzisyonunu bul
   int    pos =StringFind(path,"\\MQL5\\");
//--- altdizgi bulunamadı - hata
   if(pos<0)
      return(NULL);
//--- "\MQL5" konumunu atla
   pos+=5;
//--- fazlalık '\' sembollerini atla
   while(StringGetCharacter(path,pos+1)=='\\')
      pos++;
//--- bu bir kaynaksa veri yolunu MQL5 dizinine göre al
   if(StringFind(path,"::",pos)>=0)
      return(StringSubstr(path,pos));
//--- ilk MQL5 alt-dizini için bir ayraç bul (örneğin, MQL5\Indicators)
//--- bulunamazsa MQL5 dizinine göre veri yolun dönüş yap
   if((pos2=StringFind(path,"\\",pos+1))<0)
      return(StringSubstr(path,pos));
//--- alt-dizine göre veri yoluna dönüş yap (örneğin, MQL5\Indicators)
   return(StringSubstr(path,pos2+1));
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double& price[])
  {
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Kaynak değişkenleri

Kaynaklar, kaynak değişkenleri ile bildirilebilir ve uygun tipte bir değişkenmiş gibi kullanılabilir. Bildirim biçimi:

```
#resource kaynak_dosyasının_adresi as kaynak_değişkeninin_tipi kaynak_değişkeninin_isı
```

Bildirim örnekleri:

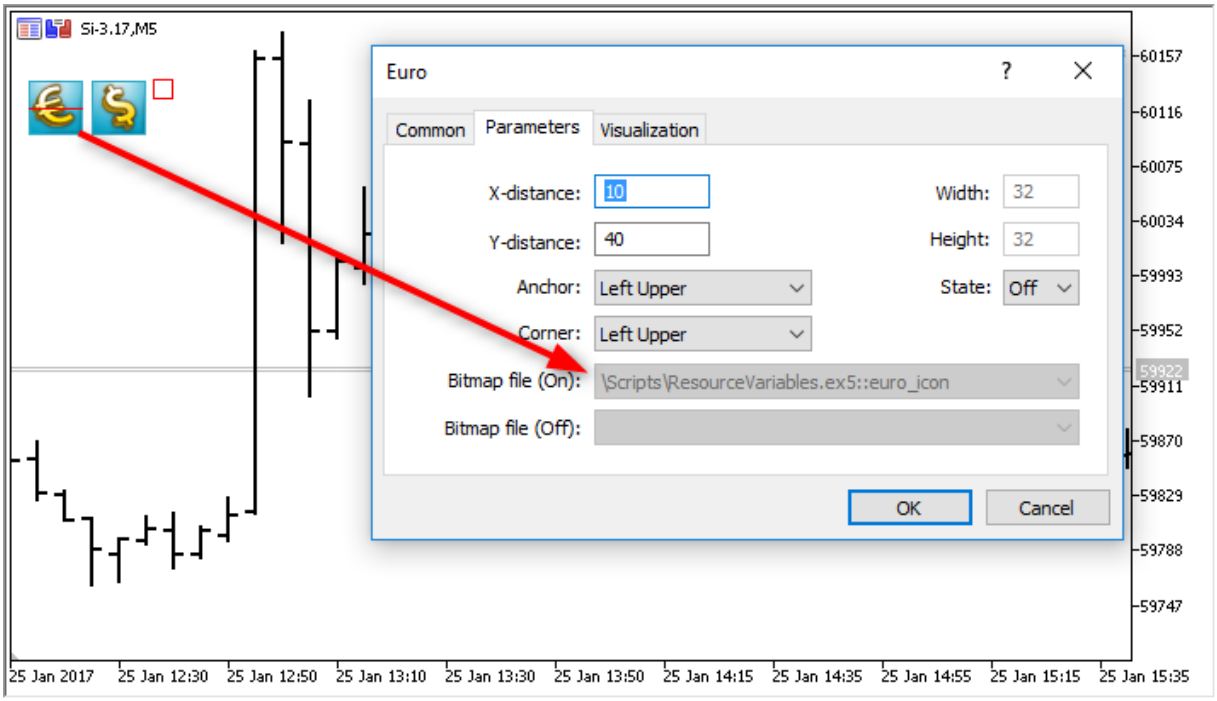
```
#resource "data.bin" as int ExtData[]             // data.bin dosyasındaki veriyi içe
#resource "data.bin" as MqlRates ExtData[]        // data.bin dosyasının verilerini iç
//--- dizgiler
```

```
#resource "data.txt" as string ExtCode            // data.bin dosyasının verilerini ic
//--- grafiksel kaynaklar
#resource "image.bmp" as bitmap ExtBitmap[]       // BMP dosyasının biteşlem verisini
#resource "image.bmp" as bitmap ExtBitmap2[][]    // BMP dosyasının biteşlem verisini
```

Bu gibi bildirimlerde, kaynak verisi sadece otomatik adresleme değişkeni ile adreslenebilir ve "::<kaynak ismi>" çalışmaz.

```
#resource "\\Images\\euro.bmp" as bitmap euro[][]
#resource "\\Images\\dollar.bmp"
//+------------------------------------------------------------------+
//|  Kaynak kullanımıyla OBJ_BITMAP_LABEL nesnesinin oluşturulması  |
//+------------------------------------------------------------------+
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);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- görselin çıktı boyutu [genislik, yükseklik] euro kaynağında depolanır
   Print(ArrayRange(euro,1),", ",ArrayRange(euro,0));
//--- euro kaynağındaki görseli değiştir - ortasına yatay kırmızı şerit çiz
   for(int x=0;x<ArrayRange(euro,1);x++)
      euro[ArrayRange(euro,1)/2][x]=0xFFFF0000;
//--- kaynak değişkenini kullanarak grafiksel kaynak oluştur
   ResourceCreate("euro_icon",euro,ArrayRange(euro,1),ArrayRange(euro,0),0,0,ArrayRang
//--- Euro grafiksel etiket nesnesini oluştur, euro_icon kaynağındaki görsel bu nesne
   Image("Euro","::euro_icon",10,40);
//--- kaynağı uygulamak için başka bir yöntem, bunun üstüne çizemiyoruz
   Image("USD","::Images\\dollar.bmp",15+ArrayRange(euro,1),40);
//--- daha önce euro kaynak değişkeni ile bildirimi yapıldığı için euro.bmp kaynağını
   Image("E2","::Images\\euro.bmp",20+ArrayRange(euro,1)*2,40); // çalıştırma zamanı
  }
```

Betik çalıştırma sonucu - üç nesnedan sadece ikisi (OBJ_BITMAP_LABEL nesneleri) oluşturulabildi. İlk nesnenin görselinde, ortada kırmızı bir şerit var.



Kaynakların kullanılmasının bir diğer avantajı da, kaynağın EX5 dosyasına eklenmeden (derleme işleminden önce) otomatik olarak sıkıştırılmasıdır. Yani, kaynak değişkenleri kullanarak tüm gerekli verileri doğrudan EX5 dosyasına ekleyebilirsiniz. Bu sayede geleneksel yazım tekniğiyle yazdığınız MQL5 programlarına göre çok daha küçük boyutlu dosyalar oluşturabilirsiniz.

Kaynak değişkenlerin kullanımı özellikle Mağaza bölümünde yayınlanacak ürünler için uygundur.

Özellikler

- TÖzel *bitmap* kaynak değişkeni tipi, kaynağın bir görsel olduğu bilgisini derleyiciye verir. Bu gibi değişkenler uint tipini alırlar.
- *bitmap* tipli kaynak değişkeni dizisi iki boyutlu olmalıdır. Bu durumda dizi boyutları şu şekilde tanımlanır [görsel_yüksekliği][görsel_genişliği]. Dizi tek boyutlu olarak belirtilmişse, dizinin elemansayısı görsel_yüksekliği*görsel_genişliği şeklinde hesaplanır.
- 24-bit'lik bir görsel indirilirken, alfa kanalı bileşeni görselin tüm pikselleri için 255 olarak ayarlanır.
- Alfa kanalı olmayan 32-bit'lik bir görsel indirilirken, alfa bileşeni görselin tüm pikselleri için 255 olarak ayarlanır.
- Alfa kanalı olan 32-bit'lik bir görsel indirilirken, pikseller herhangi bir şekilde değiştirilmez.
- Kaynak dosya boyutu 128 Mb 'tan daha büyük olamaz.
- BOM (başlık) varlığında, dizgi dosyaları için otomatik kodlama tespiti gerçekleştirilir. BOM yoksa, kodlama dosya içerikleri ile tanımlanır. sadec ANSI, UTF-8 ve UTF-16 dosya kodlamaları desteklenir. Dosyadan veri okunurken tüm dizgiler Unicode'a dönüştüülür.

OpenCL programları

Bazı programların geliştirilmesi dizgi kaynak değişkenlerin kullanım imkanını artırabilir. Örneğin, ayrı bir CL dosyasında bir OpenCL programı yazabilir ve bunu MQL5 program kaynağına bir dizgi olarak ekleyebilirsiniz.

```
#resource "seascape.cl" as string cl_program
```

```
...
int context;
if((cl_program=CLProgramCreate(context,cl_program)!=INVALID_HANDLE)
  {
   //--- sonraki eylemleri bir OpenCL programıyla gerçekleştir
  }
```

Bu örnek, herhangi bir *cl_programı* kaynak değişkeni olarak kullanılmadan da yazılabilirdi ama bütün kodu büyük bir dizgi şeklinde yazmanız gerekirdi.

Ayrıca bakınız

ResourceCreate(), ResourceSave(), PlaySound(), ObjectSetInteger(), ChartApplyTemplate(), Dosya Fonksiyonları

İçe Aktarılmış Fonksiyonların Çağrılması

Bir MQL5 programının çalıştırılması sırasında fonksiyonları içe aktarmak amacıyla, müşteri terminali erken bağlama yöntemini kullanır. Yani bir program, içe aktarılan bir fonksiyonu çağırmışsa, karşılık gelen modül (ex5 veya dll) programla birlikte yüklenir. MQL5 ve DLL kütüphaneleri, çağıran modülün iş parçacığında çalıştırılır.

Modülün tam isminin (*Drive:\Directory\FileName.Ext*) kullanılması tavsiye edilmez. MQL5 kütüphaneleri *terminal_dizini\MQL5\Libraries* klasöründen yüklenirler. Kütüphane bulunamamışsa, terminal bunu *terminal_dir\experts* klasöründen yüklemek için girişimde bulunur.

Sistem kütüphaneleri (DLL), işletim sisteminin kurallarına göre yüklenirler. Kütüphane önceden yüklenmişse (örneğin, başka bir uzman danışmandan veya paralel çalışan başka bir müşteri terminalinden bile yüklenmişse), çağrılar daha önceden yüklenmiş olan kütüphaneye gider. Aksi durumda şu sıra kullanılarak arama gerçekleştirilir:

1. Modülün çalıştığı ve dll dosyasını içe aktardığı dizin. Buradaki modül bir Uzman Danışman, bir script veya bir EX5 kütüphanesi olabilir;
2. terminal_veri_dizini\MQL5\Libraries dizini (TERMINAL_DATA_PATH\MQL5\Libraries);
3. MetaTrader 5 müşteri terminalinin başlatıldığı konum;
4. Sistem dizini;
5. Windows dizini;
6. Mevcut dizin;
7. Konumlar PATH sistem değişkeninde listelenir.

DLL dosyası içinde başka bir DLL kullanılıyorsa, ikinci DLL dosyasının olmaması durumunda birincisi de yüklenemez.

Bir Uzman Danışman (script veya gösterge) yüklenmeden önce, EX5 kütüphanelerinin genel listesi yeniden şekillendirilir. Bu liste Uzman Danışmandan (scriptten veya göstergeden) ve listedeki kütüphanelerden yüklenebilir olacaktır. Böylece, çok defa kullanılacak bir EX5 kütüphane modülünün bir defa yüklenmesi yeterli olacaktır. Kütüphaneler çağrıldıkları Uzman Danışmanlarda (scriptlerde ve göstergelerde) yer alan ön-tanımlı değişkenleri kullanırlar.

İçe aktarılan bir EX5 kütüphanesi, şu sıra ile aranır:

1. EX5 dosyasını içe aktaran Uzman Danışmanın (betiğin veya göstergenin) konumuna göre ayarlanan dizin;
2. terminal_dizini\MQL5\Libraries konumu;
3. Tüm MetaTrader 5 müşteri terminallerinin genel klasöründeki MQL5\Libraries konumu (Common\MQL5\Libraries).

Bir MQL5 programında DLL dosyasından içe aktarılan fonksiyonlar Windows API çağrı koşullarını teyit etmelidir. İşlemi teyit etmek amacıyla, C veya C++ ile yazılmış programların kaynak kodları içinde, Microsoft(r) derleyicilerine özel __stdcall anahtar sözcüğünü kullanın. İşlem koşulları şunlarla karakterizedir:

- çağrıyı yapan (bizim durumumuzda bu bir mq5-programı), parametreleri uygun bir şekilde kullanabilmek için, çağrılan (DLL'den içe aktarılan) fonksiyonun prototipini görmelidir;
- çağrıyı yapan (bizim durumumuzda bu bir mq5-programı), parametreleri ters sıra ile girmelidir, sağdan sola - içe aktarılan fonksiyon, geçirilen parametreleri bu sıra ile okur;

- Açık şekilde referansla geçirilmesi gerekenler (bizim durumumuzda dizgiler) haricindeki parametreler değer ile geçirilir,
- içe aktarılan fonksiyon, geçirilen parametreleri okuyarak yığını temizler.

İçe aktarılan fonksiyonun prototipi tarif edilirken, varsayılan parametreler kullanılabilir.

Eğer söz konusu kütüphane yüklenemiyorsa veya DLL kullanımına izin verilmiyorsa veya içe aktarılan fonksiyon bulunamamışsa, Uzman Danışman Günlükte verilecek uygun bir mesaj ile çalışmasını durduracaktır - "Uzman Danışman durduruldu". Bu durumda Uzman Danışman sonlandırılıncaya kadar çalışmayacaktır. Bununla birlikte, yeni bir derleme sonucunda veya özellikler penceresi açılıp "Tamam" tuşuna basıldığında yeniden başlatılabilir.

Parametrelerin Geçirilmesi

Açık bir şekilde referans ile geçirilmeleri belirtilmedikçe tüm basit tipler değer ile geçirilir. Bir dizgi DLL den aktarılan bir fonksiyona geçirildiğinde, kopyalanan dizginin önbellek adresi geçirilir; eğer dizgi referans ile geçirilmişse, dizginin önbellek adresi kopyalanmadan geçirilir.

Dinamik dizileri, dizgileri, sınıfları, diğer karmaşık yapıları ve numaralandırılmış nesnelerin statik veya dinamik dizilerini içeren yapılar, içe aktarılan bir fonksiyona parametre olarak geçirilemez

Bir diziyi DLL fonksiyonuna geçirirken, her zaman veri tamponunun başlangıç adresi geçirilir (AS_SERIES bayrağına bakılmaksızın). DLL içerisindeki fonksiyonlar AS_SERIES bayrağı ile kullanılamazlar, geçirilen diziler statiktir ve büyüklükleri tanımlanmamıştır; dizi büyüklüğünü tanımlamak için fazladan bir parametre kullanılmalıdır.

Çalışma Zamanı Hataları

Müşteri terminalinin idari alt sistemi, bir MQL5 programının çalışması sırasında ortaya çıkan hata kodunu kaydetme imkanına sahiptir. Bunun için, _LastError ön-tanımlı değişkeni, çalıştırılabilir her MQL5 programında bulunmaktadır.

OnInit fonksiyonu başlatılmadan önce, _LastError değişkeni sıfırlanır. Hesaplamada veya içsel fonksiyon çağrılarında bir hata gerçekleşmesi durumunda, _LastError değişkeni karşılık gelen hata kodunu alır. Bu değişken tarafından saklanan değer GetLastError() fonksiyonu ile elde edilebilir.

Bir programın aniden sonlandırılması durumunda çeşitli kritik hatalar mevcuttur:

- sıfır ile bölme
- dizi sınırları dışına çıkılması
- nesne işaretçisinin hatalı kullanımı

Alım-Satım Stratejilerinin Sınanması

Otomatik alım-satım fikri, bir alım-satım robotunun haftanın 7 günü 24 saat aralıksız çalışabilmesinden kaynaklanmaktadır. Robotlar yorulmaz, şüpheye düşmez ve korkmaz, tüm psikolojik problemlerden uzaktır. Sadece alım-satım kurallarının belirlenmesi ve bir algoritmada uygulanması yeterlidir; ve robot, yorulmadan çalışmak için hazırdır. Ama ilk önce, şu iki koşulun sağlandığından emin olmanız gerekir:

- Uzman Danışman, alım-satım sisteminin kurallarına bağlı olarak alım-satım işlemleri gerçekleştirir;
- Uzman Danışman içine uygulanan alım-satım stratejisi geçmiş veri üzerinde karlılık gösterir.

Bu iki sorunun cevabını alabilmek için, MetaTrader 5 müşteri terminaline eklenmiş olan Strateji Sınama aracına ihtiyaç duyarız.

Bu bölüm, programların Strateji Sınayıcı içinde sınanması ve optimizasyonu ile ilgili özellikleri kapsamaktadır:

- Strateji Sınayıcı Dahilindeki Fonksiyon Kısıtlamaları
- Tik Oluşturma modları
- Makas Değerinin Simülasyonu
- Sınama sırasında gerçek tiklerin kullanımı
- Müşteri Terminalinin Global Değişkenleri
- Sınama sırasında göstergelerin hesaplanması
- Sınama sırasında Geçmiş Verinin Yüklenmesi
- Çoklu-Döviz Sınaması
- Strateji Sınama Aracında Zamanın Simülasyonu
- Sınama içinde Grafiksel Nesneler
- Strateji Sınayıcıda OnTimer() Fonksiyonu
- Strateji Sınayıcıda Sleep() Fonksiyonu
- Matematiksel Hesaplamalardaki Optimizasyon Problemleri için Strateji Sınama aracının Kullanılması
- "Sadece Açılış fiyatları" modunda Çubukların Senkronizasyonu
- Sınayıcı içinde IndicatorRelease() Fonksiyonu
- Sınama Aracında Olay İşleme
- Sınama Temsilcileri
- Terminal ve Sınama Temsilcisi arasında Veri Değişimi
- Tüm Müşteri Terminallerinin Ortak Klasörünün Kullanılması
- DLL Dosyalarının Kullanımı

MQL5 Bulut Ağında bellek ve disk alanı sınırları

MQL5 Bulut Ağında yürütülen optimizasyonlar için şu sınırlama mevcuttur: Uzman Danışman, diske 4 GB'tan fazla bilgi yazmamalı veya 4 GB'tan fazla RAM kullanmamalıdır. Sınır aşılırsa ağ temsilcisi

hesaplamayı doğru bir şekilde tamamlayamaz ve dolayısıyla sonucu alamazsınız. Ancak yine de hesaplamalara harcanan tüm süre için ücretlendirilirsiniz.

Her optimizasyon geçişinden bilgi almanız gerekiyorsa, çerçeveleri diske yazmadan gönderin. MQL5 Bulut Ağında hesaplamalar sırasında Uzman Danışmandaki dosya işlemlerini kullanmaktan kaçınmak için aşağıdaki kontrolü kullanabilirsiniz:

```
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(...);
   ...
  }
```

Strateji Sınayıcı Dahilindeki Fonksiyon Kısıtlamaları

Müşteri terminalinin bünyesinde yer alan Strateji Sınayıcı içinde, bazı fonksiyonlar için işlem kısıtlamaları bulunur.

Comment(), Print() ve PrintFormat() Fonksiyonları

Alım-satım robotları optimize edilirken Comment(), Print() ve PrintFormat() fonksiyonları performansı artırmak için çalıştırılmaz. Bu fonksiyonların OnInit() işleyicisinin içinde kullanılması bu kısıtlamalara bir istisna oluşturur. Böylece hataların neden kaynaklandığını kolayca bulabilirsiniz.

Alert(), MessageBox(), PlaySound(), SendFTP, SendMail(), SendNotification(), WebRequest() Fonksiyonları

Alert(), MessageBox(), PlaySound(), SendFTP(), SendMail(), SendNotification() ve WebRequest() fonksiyonları dış dünya ile iletişim için tasarlanmıştır ve Strateji Sınayıcı içinde çalıştırılmazlar.

Tik Oluşturma modları

Uzman Danışman, MQL5 dilinde yazılan ve her seferinde bazı dışsal olaylara yanıt olarak çalışan bir programdır. Uzman danışman, her ön-tanımlı olay için karşılık gelen bir fonksiyon (olay işleyici) içerir.

NewTick olayı (fiyat değişimi) bir Uzman için ana olaydır, bu nedenle, Uzman Danışmanı sınayabilmek için bir tik dizisine gereksinim duyarız. MetaTrader 5 müşteri terminalinin Strateji Sınama aracında uygulanan 3 adet tik oluşturma modu bulunmaktadır:

- Every tick

- 1 dakikalık çubuklarla OHLC fiyatları (1 Minute OHLC)
- Open prices only

En temel ve en detaylı olanı "Her tik" (Every tick) modudur, diğer iki mod ise temel modun basitleştirilmiş halidir ve "Her tik" moduyla karşılaştırmalı olarak açıklanacaklardır. Aralarındaki farkı anlamak amacıyla üç modun tamamını düşünün.

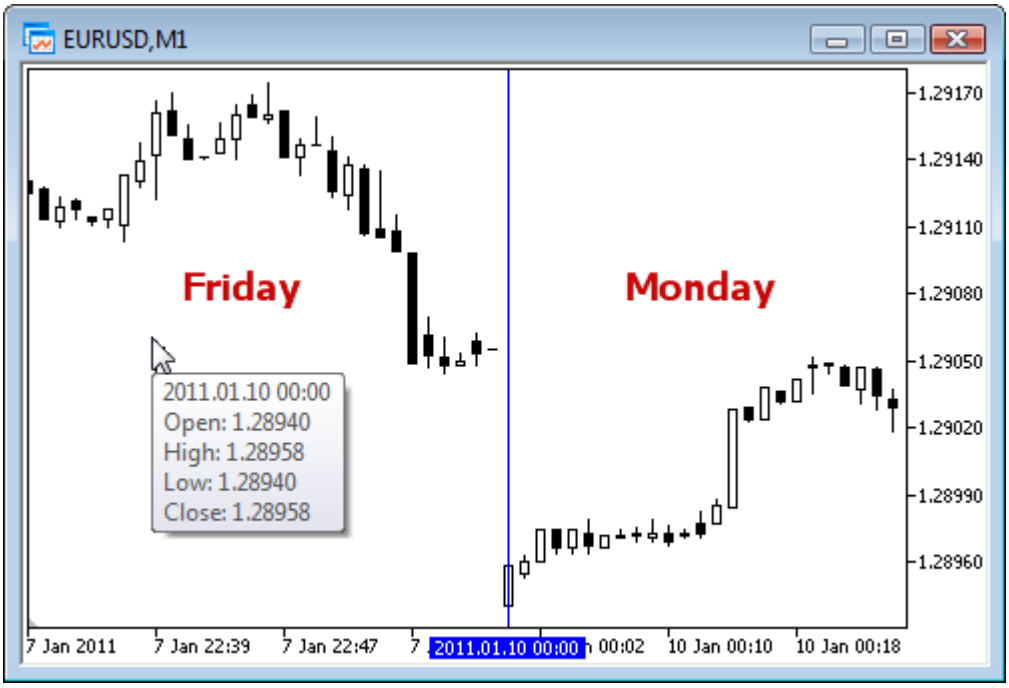
"Her tik"

Finansal enstrümanlara dair geçmiş fiyat verileri, dakikalık çubuklar şeklinde paketlenmiş olarak sunucudan MetaTrader 5 müşteri terminaline aktarılır. İsteklerin yapılmasına ve istenen zaman aralığının yapımına dair detaylı bilgiyi, MQL5 Referans dosyasının Veri Erişiminin Düzenlenmesi bölümünde bulabilirsiniz.

Fiyat geçmişinin en küçük elemanı dakikalık çubuktur, bundan dört fiyat değerine dair verileri alabilirsiniz:

- Open - dakikalık çubuğun açılış fiyatı;
- High - dakikalık çubuk içinde gerçekleşmiş en yüksek fiyat değeri;
- Low - dakikalık çubuk içinde gerçekleşmiş en düşük fiyat değeri;
- Close - çubuğun kapanış fiyatı.

Yeni dakikanın çubuğu, yeni dakikanın başlamasıyla (toplam saniye sayısının 0 olmasıyla) değil, en az bir puanlık bir fiyat değişimini içeren yeni bir tikin gelmesiyle açılır. Aşağıdaki resim, yeni işlem haftasının 2011.01.10 00:00 açılış zamanına sahip ilk bir dakikalık çubuğunu göstermektedir. Cuma ve Pazartesi günleri arasındaki fiyat boşluğu, kurların gelen haberlere bağlı olarak, hafta-sonu da dalgalanmaları nedeniyle sık karşılaşılan bir durumdur.

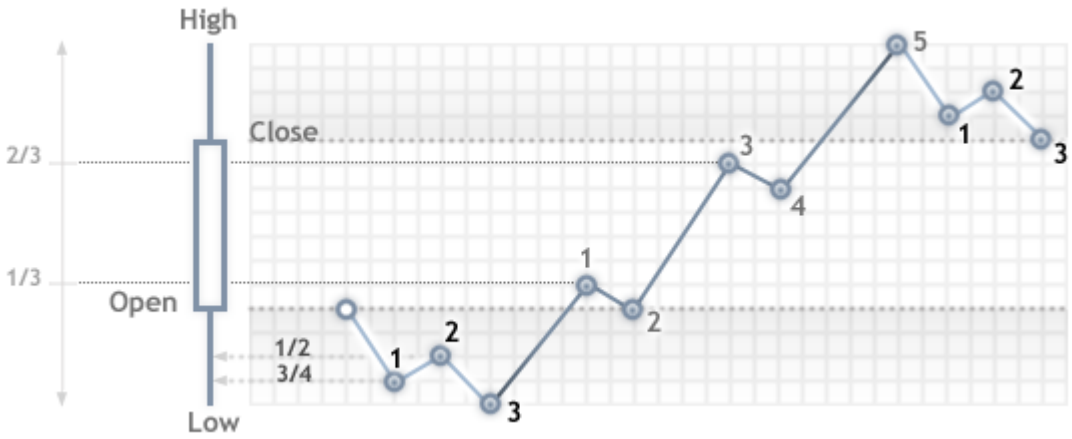


Bu çubuğun 10 ocak 2011 günü 00 saatinde ve 00 dakikasında başladığını biliyoruz ama saniye hakkında hiç bir bilgimiz yok. 00:00:12'de , 00:00:36'da (yeni günün başlangıcından 12 veya 36 saniye sonra) veya o dakika içerisindeki herhangi bir anda açılmış olabilir. Ama EURUSD Open (açılış) fiyatının, açılış anında 1.28940 olduğunu bilebiliriz.

Ayrıca söz konusu dakikalık çubuğun kapanışına karşılık gelen tik fiyatının, saniye içerisindeki geliş anını tam olarak bilmiyoruz. Sadece bir şey biliyoruz : bir dakikalık çubuğun son Close (kapanış) fiyatını. Bu dakika için fiyat 1.28958 seviyesinde. High ve Low fiyatlarının oluşma anlarını da bilmiyoruz ama maksimum ve minimum fiyatların sırasıyla 1.28958 ve 1.28940 seviyelerinde olduğunu biliyoruz.

Alım satım stratejisini sınamak amacıyla, Uzman Danışmanın çalışmasını simüle edebileceğimiz, tiklerden oluşan bir diziye ihtiyaç duyarız. Bu şekilde, her bir dakikalık çubuk için, fiyatın kesinlikle uğramış olduğu **4 kontrol noktasını** biliriz. Bir çubuğun dört tik içerdiğini biliyorsak, bu bilgi bir sınama gerçekleştirmek için yeterli olacaktır, diğer taraftan tik hacmi genellikle dördün üzerindedir.

İşte bu nedenle, Open, High, Low ve Close fiyatlarının arasında, ek kontrol noktaları oluşturulması gerekir. "Her tik" tik oluşturma modunun temel ilkesi MetaTrader 5 Terminalinin Strateji Sınama Aracının Tik Oluşturma Algoritması kısmında açıklanmıştır, aşağıdaki şekilde temsil edilmektedir.



"Her tik" modunda sınama yapılırken, Uzman Danışmanın OnTick() fonksiyou her kontrol noktasında çağrılacaktır. Her kontrol noktası oluşturulan diziden gelen bir tiktir. Uzman Danışman, simüle edilen tik zamanını ve fiyatını çevirim-içi çalışmalarda olduğu gibi alacaktır.

> **Önemli:** "Her tik", en gerçekçi sınama modudur ama aynı zamanda en çok zaman harcayandır. Alım-satım stratejilerinin büyük çoğunluğu için, başlangıç olarak diğer iki sınama modunun kullanımı, genellikle yeterlidir.

"1 Dakikalık OHLC"

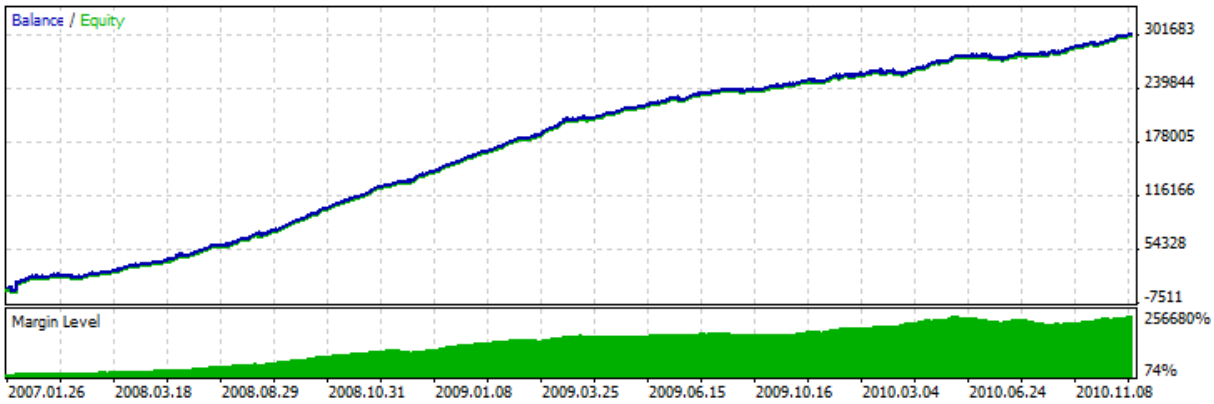
"Her tik" modu, üç sınama modu arasında en gerçekçi olandır ama aynı zamanda en yavaşıdır. OnTick() işleyicisinin çalışması her tikte gerçekleşir ve tik hacmi oldukça büyük olabilir. Çubuk içindeki minimal fiyat hareketlerinin önem taşımadığı bir strateji için, daha hızlı ve daha kaba bir simülasyon modu bulunmaktadır - "1 dakikalık OHLC".

"1 dakikalık OHLC" modunda, tik dizisi sadece **dakikalık çubukların OHLC fiyatlarından** oluşur; oluşturulan kontrol noktalarının sayısı - ve sınama süresi - anlamlı ölçüde azaltılmıştır. OnTick () fonksiyonu, dakikalık çubukların OHLC fiyatlarından oluşan tüm kontrol noktalarında çalıştırılır.

Open, High, Low ve Close fiyatlarının arasında ek tiklerin oluşturulmasının reddedilmesi; Open fiyatının belirlendiği andan itibaren, fiyatların geliştirilmesinde katı bir determinizm görüntüsü ortaya

çıkaracaktır. Bu, test edilen bakiyenin yukarı yönlü grafiğini gösteren bir "Sınama Kasesi", oluşturulmasını mümkün kılar.

Böyle bir Kase örneği Code Base - Grr-al linkinde bulunabilir.



Resimde bu Uzman Danışmanın etkileyici grafiği gözükmektedir. Nasıl elde edildi? Bir dakikalık çubuk için dört ayrı fiyat biliyoruz ve yine biliyoruz ki Open fiyatı bunların ilkiyken Close fiyatı ise sonuncusu. Bunların arasında High ve Low fiyatlarımız var ve bunların oluşma sıraları belirsiz ama High fiyatının Open fiyatından büyük veya ona eşit olduğunu (ve Low fiyatının Open fiyatından küçük veya ona eşit olduğunu) biliyoruz.

Open fiyatının alınma anını belirlemek, ardından o anda hangi fiyatı - High veya Low - aldığımızı belirlemek için sonraki tikleri analiz etmek yeterli olacaktır. Fiyat açılış (Open) fiyatının altındaysa, bu Low fiyatına sahip olduğumuz anlamına gelir ve bu tik üzerinde alım yaparız, bir sonraki tik High fiyatına karşılık gelir burada alış pozisyonunu kapatır ve satış için yeni pozisyon açarız. Bir sonraki (son) tik, Close fiyatıdır ve burada da satış pozisyonunu kapatırız.

Açılıştan sonra aldığımız tik açılçış fiyatından büyükse, bu işlem dizisini ters yönde uygularız. Bu "hile" modunda bir dakikalık çubuğu işleyin, ardından bir sonraki için bekleyin.

Böyle bir Uzman Danışmanı geçmiş veriyle sınarken herşey düzgün gider ama bir kez çevirim içi çalıştırdığımızda, asıl gerçek ortaya çıkar - bakiye çizgisi istikrarlıdır ama aşağı doğru yönelmiştir. Bu hileyi ortaya çıkarmak için, Uzman Danışmanı "Her Tik" modunda çalıştırmamız gerekir.

> **Not:** Bir uzman Danışmanın test sonuçları, kaba sınama modlarında ("1 dakikalık OHLC" ve "Sadece Açılış Fiyatları") çok iyi görünüyorsa "Her Tik" modunda da Sınama gerçekleştirdiğinizden emin olun.

"Sadece Açılış Fiyatları"

Bu modda tikler, sınama için seçilen zaman aralığında OHLC fiyatlarının temelinde oluşturulur. Uzman Danışmanın OnTick() fonksiyonu, sadece çubuğun başlangıcındaki Open fiyatında çalışır. Bu özellik nedeniyle, stop (durdurma) ve bekleyen emir seviyeleri, belirtilenden farklı bir fiyat ile tetiklenebilir (özellikle test sırasında büyük periyotlar kullanılıyorsa). Bunun yerine Uzman Danışman için hızlı bir ön değerlendirme testi yapabilme olanağımız bulunmaktadır.

W1 ve MN1 periyotları, "Sadece Açılış Fiyatları" modu için birer istisnadır: bu periyotlar için tikler, haftalık veya aylık değil, günlük OHLC fiyatları ile oluşturulur.

EURUSD H1 sembolünde, "Sadece Açılış Fiyatları" modunda bir Uzman Danışmanı test ettiğimizi düşünelim. Bu durumda tiklerin (kontrol noktalarının) toplam sayısı, sınamanın yapıldığı zaman aralığındaki bir-saatlik çubukların sayısının 4 katı olacaktır. Ama **OnTick() işleyicisi, sadece bir-saatlik çubuğun açılışında çağrılacaktır**. Doğru sınama için istenen kontroller, tiklerin geri kalanı (Uzman Danışmandan gizlenenler) üzerinde gerçekleşir.

- Marjin (teminat) gerekliliklerinin hesaplanması;
- Zarar Durdur ve Kar Al seviyelerinin tetiklenmesi;
- Bekleyen emirlerin tetiklenmesi;
- Zaman-aşımına uğramış bekleyen emirlerin kaldırılması.

Herhangi bir açık pozisyon veya bekleyen emir yoksa, gizli tiklerin üzerinde bu kontrolleri gerçekleştirmemiz gerekmez, bu şekilde ciddi bir hız artışı sağlanabilir. "Sadece Açılış Fiyatları" modu, işlemleri sadece açılış fiyatından gerçekleştiren ve ve StopLoss, TakeProfit ve bekleyen emir kullanmayan stratejiler için oldukça uygundur. Bu tip stratejiler için gereken test geçerliliği korunur.

Standart paketteki Uzman Danışmanlardan, her modda test edilebilecek olan Moving Average Uzman Danışmanını örnek olarak alalım. Bu Uzman Danışmanın mantığı, tüm kararların çubuğun açılışında verilmesi, işlemlerin hemen uygulanması ve bekleyen emir kullanılmaması üzerine kuruludur.

Bu uzman için EURUSD H1 sembolünde, 2010.09.01 ile 2010.12.31 aralığında bir test düzenleyip sonuç grafiklerini karşılaştıralım. Aşağıdaki resim, üç modun her biri için test raporundaki bakiye grafiğini göstermektedir.

Sizin de görebileceğiniz gibi, farklı modlardaki tüm grafikler, standart paketteki Moving Average uzmanı için aynıdır.

"Sadece Açılış Fiyatları" modunda bir takım kısıtlamalar bulunmaktadır:

- Rassal Gecikme uygulama modunu kullanamazsınız.
- Test edilen Uzman Danışman için, sınama/optimizasyon için kullanılandan daha küçük bir zaman-aralığı verisine erişemezsiniz. Örneğin, sınamayı/optimizasyonu H1 periyodu üzerinde

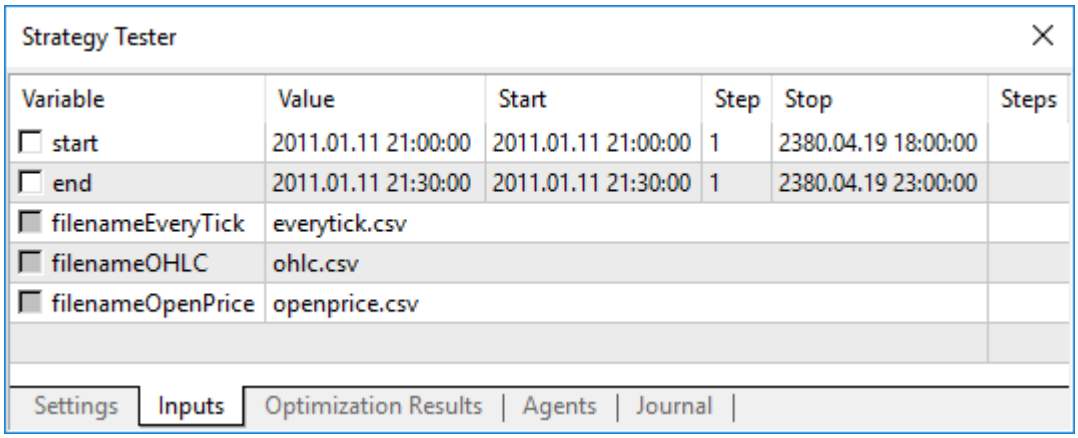
gerçekleştirirseniz, H2, H3, H4 vb. periyotların verilerine erişebilirsiniz ama M30, M20, M10 vb. verilerine erişemezsiniz. Ayrıca, erişilmek istenen daha yüksek zaman-aralıkları, sınama için kullanılan zaman-aralığının katı olmalıdır. Örneğin, M20 üzerinde bir sınama gerçekleştiriyorsanız, M30 verisine erişemezsiniz ama H1 verisine erişmeniz mümkündür. Bu kısıtlamalar, sınama sürecinde oluşturulan çubukların katı-olmayan veya onlardan daha düşük periyotlu çubuklardan veri alınmasının imkansızlığı ile ilişkilidir.

- Diğer zaman-aralıklarının verilerindeki erişim kısıtlamaları, Uzman Danışman tarafından kullanılan diğer sembollere de uygulanır. Bu durumda, her bir sembol için yapılacak kısıtlama, sınama/optimizasyon sırasında erişilen ilk zaman-aralığına bağlı olacaktır. EURUSD H1 üzerinde yapılan bir sınama sırasında, Uzman Danışmanın GBPUSD M20 verisine erişim sağladığını düşünün. Bu durumda Uzman Danışman, EURUSD H1, H2, vb. verilerine de, GBPUSD M20, H1, H2, vb. verilerine eriştiği gibi erişebilecektir.

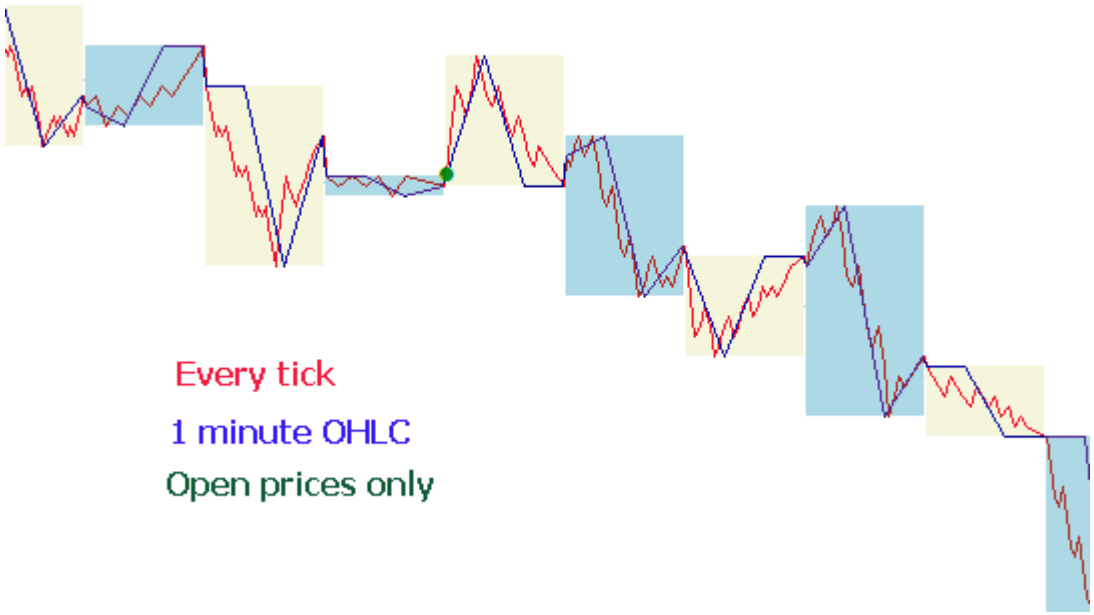
Not: "Sadece Açılış Fiyatları" modu, en kısa test süresine sahiptir fakat tüm alım-satım stratejileri için uygun değildir. İstenen sınama modunu, alım-satım stratejisinin karakteristiklerine göre seçiniz.

Tik oluşturma bölümünü sonuca bağlamak amacıyla, EURUSD sembolü ve M15 periyodu için, 2011.01.11 21:00:00 - 2011.01.11 21:30:00 aralığında, farklı tik oluşturma modlarının görsel karşılaştırmasını inceleyelim.

Tikler, WriteTicksFromTester.mq5 Uzman Danışmanı kullanılarak farklı dosyalara yazılmışlardır ve bu dosya isimlerinin sonları filenameEveryTick, filenameOHLC ve filenameOpenPrice giriş parametreleri ile belirlenmiştir.



Üç tik dizisini içeren üç dosyayı almak için, Uzman Danışman, karşılık gelen üç ayrı modda ("Her tik", "1 dakikalık OHLC" ve "Sadece Açılış Fiyatları") tekil olarak (üç defa) çalıştırılır. Ardından, bu üç dosyadaki veriler TicksFromTester.mq5 göstergesi ile çizelge üzerinde görüntülenir. Gösterge kodları bu yazıya eklenmiştir.



Varsayılan olarak, MQL5 dilindeki tüm dosya işlemleri, "dosya güvenlik-ortamı" (sandbox) içinde gerçekleştirilir; sınama sırasında Uzman Danışman, sadece kendi "dosya güvenlik-ortamına" erişebilir. Göstergenin ve Uzman Danışmanın sınama sırasında aynı klasördeki dosyalarla çalışabilmesi için, FILE_COMMON bayrağını kullandık. Uzman Danışmandan bir kod örneği:

```
//--- dosyayı aç
   file=FileOpen(filename,FILE_WRITE|FILE_CSV|FILE_COMMON,";");
//--- dosya tanıtıcı değerini kontrol et
   if(file==INVALID_HANDLE)
     {
      PrintFormat("Yazılacak %s dosyasının açılmasında hata. Hata kodu=%d",filename,Ge
      return;
     }
   else
     {
      PrintFormat("Dosya, %s klasöründe oluşturulacak",TerminalInfoString(TERMINAL_CO
     }
```

Gösterge içindeki veriyi okumak için de yine FILE_COMMON bayrağını kullandık. Bu, gerekli dosyaları bir klasörden diğerine, el yordamıyla taşımaktan kaçınmamızı sağladı.

```
//--- dosyayı aç
   int file=FileOpen(fname,FILE_READ|FILE_CSV|FILE_COMMON,";");
//--- dosya tanıtıcı değerini kontrol et
   if(file==INVALID_HANDLE)
     {
      PrintFormat("%s dosyasının okuma amacıyla açılmasında hata. Hata kodu=%d",fname,
      return;
     }
   else
     {
```

```
      PrintFormat("Dosya, %s klasöründe açılacak",TerminalInfoString(TERMINAL_COMMONDA
     }
```

Makas Değerinin Simülasyonu

Satış ve Alış fiyatları arasındaki fark, makas (spread) olarak adlandırılır. Sınama sırasında makas değeri modellenmez, geçmiş veriden veriden alınır. Geçmiş verideki makas değeri sıfıra eşit veya sıfırdan küçükse, bilinen son makas değeri sınama temsilcisi tarafından kullanılır.

Strateji Sınayıcıda makas değeri her zaman kayan noktalı şekilde düşünülür. Bu SymbolInfoInteger(symbol, SYMBOL_SPREAD_FLOAT) çağrısının her zaman true dönüşü yapmasından kaynaklanır.

Ayrıca, geçmiş veriler, tik değerlerini ve alım-satım hacimlerini de içerirler. Verileri depolamak ve almak için, özel MqlRates yapısını kullanırız:

```
struct MqlRates
  {
   datetime time;         // Periyot başlangıç zamanı
   double   open;         // Açılış fiyatı
   double   high;         // Periyodun en yüksek fiyatı
   double   low;          // Periyodun en düşük fiyatı
   double   close;        // Kapanış fiyatı
   long     tick_volume;  // Tik hacmi
   int      spread;       // Makas
   long     real_volume;  // Alım-satım hacmi
  };
```

Sınama sırasında gerçek tiklerin kullanımı

Gerçek tiklerle yapılan sınama ve optimizasyonlar piyasa koşullarına çok yakındır. Dakikalık verilere göre oluşturulan tiklere ek olarak aracı kurumca toplanan gerçek tik verileri de kullanılabilir. Bunlar borsalardan ve likidite sağlayıcılardan gelen tiklerdir.

En tutarlı sınama sonuçlarını elde etmek için, gerçek tikler modunda dakikalık çubuklar da kullanılır. Çubuklar tik verisini kontrol etmeye ve düzeltmeye yarar. Böylece, sınayıcıda ve terminalde gösterilen çizelgelerin farklı olması önlenir.

Sınayıcı tik verilerini dakikalık çubuklarla karşılaştırır: tik değerleri çubuğun Yüksek/Düşük verileri arasında olmalıdır, ilk ve son tikler Açılış/Kapanış verileriyle örtüşmelidir. Ayrıca hacim değeri de karşılaştırılır. Uyumsuzluk tespit edilirse o dakikaya dair tüm veriler geçersiz sayılır ve bunların yerine, oluşturulan sanal tikler kullanılır ("Her Tik" modunda olduğu gibi).

Sınayıcı, sembol geçmişinde tik verisi bulunmayan dakikalık çubuklar için "Her tik" modunda tik oluşturur. Bu sayede aracı kurum verilerinin yetersiz olması durumunda çizelge doğru çizilebilir.

Sembol geçmişinde dakikalık veriler bulunmuyorsa ama ilgili dakika için tik verileri mevcutsa, veriler sınayıcıda kullanılabilir. Örneğin, borsalarda sembol çiftleri Son fiyatlar kullanılarak oluşturulur. Sunucudan gelen veriler Alış/Satış verilerini içeriyor ama Son fiyat bilgisini içermiyorsa çubuk oluşturulmaz. Dakikalık veriler olmadığı sürece sınayıcı bu tikleri kullanacaktır.

Bağlantı kayıpları veya kaynak ve terminal arasında veri iletimi sırasında oluşan hatalar vb. gibi sebeplerden ötürü, tik verileri dakikalık verilerle örtüşmeyebilir. Sınama sırasında dakikalık veriler daha güvenilir kabul edilir.

Gerçek tiklerle sınama yaparken şu bilgileri göz önünde bulundurun:

- Sınama sırasında sembolün dakikalık verileri tik verileriyle eşitlenir.
- Tikler strateji sınayıcısının sembol kasasında saklanır. Kasa boyutu 128 000 tik verisini aşamaz. Yeni tiklerin gelmesiyle, kasadaki en eski tikler silinir. Ama CopyTicks fonksiyonu ile kasa dışında kalan tik verileri alınabilir (sadece gerçek tiklerle çalışırken). Bu durumda, sınayıcının tik veri-tabanından istenen veriler terminal veri-tabanıyla tamamen aynı olur. Bu verilere dakikalık çubuklar için düzeltme uygulanmaz. Bu yüzden kullanılan tikler kasadaki tiklerden farklı olabilir.

Müşteri Terminalinin Global Değişkenleri

Sınama sırasında, müşteri terminalinin global değişkenleri de ayrıca simüle edilir ama bunlar, (F3 tuşu kullanılarak görülebilecek) terminalin mevcut global değişkenleri ile ilişkili değildirler. Yani, sınama sırasındaki terminalin global değişkenleri ile ilgili tüm işlemler, müşteri terminalinin dışında (sınama temsilcisinde) yer alırlar.

Sınama sırasında göstergelerin hesaplanması

Gerçek zamanlı modda, gösterge değerleri her tikte hesaplanır.

Strateji Sınayıcıda, göstergeler yalnızca verilere erişildiğinde, yani göstergenin arabellek değerleri istendiğinde hesaplanır. Tek istisna, belirtilen #property tester_everytick_calculate özelliğine sahip özel göstergelerdir. Bu durumda, her tik üzerinde yeniden hesaplama yapılır.

Görsel test modunda, görsel test grafiğinde doğru şekilde görüntülenmeleri için yeni tik geldiğinde tüm göstergeler koşulsuz olarak yeniden hesaplanır.

Gösterge, her tik için bir kez hesaplanır. Gösterge verileri için sonraki tüm talepler, yeni tik gelene kadar yeniden hesaplamaya yol açmaz. Bu nedenle, zamanlayıcı EventSetTimer() fonksiyonu aracılığıyla bir uzman danışmanda etkinleştirilirse, gösterge verileri, OnTimer() işleyicisinin her çağrısından önceki son tik ile istenir. Son tikte gösterge henüz hesaplanmadıysa, gösterge değerlerinin hesaplanmasına başlanır. Veriler önceden hazırlanmışsa, yeni bir yeniden hesaplama yapılmadan veriler sağlanır.

Bu nedenle, tüm gösterge hesaplamaları en fazla kaynak tasarrufu sağlayacak şekilde gerçekleştirilir - gösterge tik işaretinde zaten hesaplanmışsa, verileri 'olduğu gibi' sağlanır. Yeniden hesaplama yapılmaz.

Sınama sırasında Geçmiş Verinin Yüklenmesi

Sınanacak sembolün geçmişi, sınama süreci başlamadan önce terminal tarafından, alım-satım sunucusundan indirilerek senkronize edilir. Terminal, sembolün mevcut olan bütün geçmişini bir daha istememek üzere ilk seferde yükler. Bundan sonra sadece yeni veriler yüklenir.

Sınama temsilcisi, sınanacak sembolün geçmişini sınama başladıktan hemen sonra müşteri terminalinden alır. Eğer, test sürecinde diğer enstrümanların verileri de kullanılıyorsa (çok-dövizli bir Uzman Danışman gibi), sınama temsilcisi, verinin çağrılmasıyla birlikte müşteri terminalinden gereken

veriyi ister. Tarihsel veri terminalde mevcutsa, hemen sınama temsilcisine aktarılır. Eğer veriler mevcut değilse, terminal verileri sunucudan ister ve yükler, ardından sınama temsilcisine aktarır.

Ek enstrümanların verilerine, alım-satım işlemleri için çapraz-kurları hesaplamak amacıyla da ihtiyaç duyulur. Örneğin, EURCHF üzerinde USD karşıt döviz birimi ile bir sınama gerçekleştirirken, ilk alım-satım işlemini gerçekleştirmeden önce, sınama temsilcisi EURUSD ve USDCHF sembollerinin geçmişlerini müşteri terminalinden ister; ama strateji bu sembollerin doğrudan kullanımını içermez.

Çok-dövizli bir stratejiyi sınamadan önce, gereken tüm geçmiş verinin müşteri terminaline yüklenmesi önerilir. Bu, sınama sırasında, istenen verinin yüklenmesinin yol açabileceği gecikmelerin önlenmesine yardımcı olur. geçmişi, uygun çizelgeyi açıp, onu geçmiş veri başlangıcına kadar kaydırarak da yükleyebilirsiniz. Terminale zorla veri yüklemeye dair bir örnek de, MQL5 Referansının Veri Erişiminin Düzenlenmesi bölümünde yer almaktadır.

Sınama Temsilcileri, geçmişi terminalden paketlenmiş olarak alır. Bir sonraki sınama boyunca, sınayıcı veriyi terminalden yüklemez; istenen veri, sınayıcının bir önceki çalışmasından zaten mevcuttur.

- Sınama temsilcisi sınanacak sembolün geçmişini terminalden istediğinde, terminal geçmişi sunucudan sadece bir kere yükler. geçmiş, trafiği azaltmak için paketlenmiş şekilde yüklenir.
- Tikler ağ üzerinden gönderilmez; sınama temsilcilerinde oluşturulurlar.

Çoklu-Döviz Sınaması

Strateji Sınama Aracı, çoklu sembollerle alım-satım yapan stratejileri sınamamıza imkan tanır. Önceki platformlarda sınama işlemi sadece tek sembol için gerçekleştirildiğinden, bunlar sıklıkla çok-dövizli Uzman Danışmanlar olarak isimlendirilir. . MetaTrader 5 terminalinin Strateji Sınama Aracı içinde, alım-satım stratejilerini mevcut tüm semboller için modelleyebiliriz.

Sınama aracı, kullanılan sembollerin geçmiş verilerini, verinin çağrılmasıyla birlikte **müşteri terminalinden** (alım-satım sunucusundan değil!) otomatik olarak yükler.

Sınama temsilcisi, sınama başlangıcında, sadece göstergenin hesaplanması için gereken eksik verileri indirir (geçmiş verileri sağlamak için küçük bir farkla) D1 ve daha düşük zaman-aralıkları için, indirilen minimum geçmiş veri hacmi bir yıldır.

Yani, 2010.11.01-2010.12.01 aralığında M15 periyoduyla (1 aylık aralıkta 15 dakikalık çubuklarla) bir sınama gerçekleştiriyorsak, terminal 2010 yılının tamamı için enstrümanın geçmişini isteyecektir. Haftalık zaman aralığı içinse, yaklaşık iki yıla denk gelen (bir yıl 52 hafta içerir)100 çubukluk geçmiş veri yüklenir. Aylık zaman-aralığında sınama gerçekleştirmek için, sınama temsilcisi 8 yıllık geçmiş veri isteyecektir (12 ay x 8 yıl = 96 ay).

Yeterli sayıda çubuk yoksa, sınama işleminden önce gereken çubuk rezervini sağlamak amacıyla, **başlangıç tarihi otomatik olarak kaydırılacaktır** (geçmişten bu güne).

Sınama sırasında, sembollerle ilgili bilgilerin alınabildiği "Piyasa Gözlemi" de ayrıca simüle edilecektir.

Ön-tanımlı olarak, sınama başlangıcında Strateji Sınama aracındaki "Piyasa Gözlemi" içinde sadece bir sembol bulunur - sınamanın yapılacağı sembol. Tüm gerekli semboller, başvurulduğunda Strateji Sınama Aracının (terminalin değil!) "Piyasa Gözlemine" bağlanırlar

Çok-dövizli bir Uzman Danışmanı sınamaya başlamadan önce, istenilen sembolün terminalin Piyasa Gözleminden seçilmesi ve istenen verinin yüklenmesi gerekir. "Yabancı" bir sembolün ilk çağrısı sırasında sembolün geçmişi, terminal ve sınama temsilcisi arasında senkronize edilir. "Yabancı" sembol, sınama işleminin yapıldığı sembolden farklı olan semboldür.

"Başka bir sembolün verilene, şu durumlarda başvuru yapılır":

- Teknik gösterge fonksiyonlarının ve IndicatorCreate() fonksiyonunun, sembol/periyot üzerinde kullanımı;
- Başka bir sembolün verisi için "Piyasa Gözlemine" yapılan istek:
  1. SeriesInfoInteger
  2. Bars
  3. SymbolSelect
  4. SymbolIsSynchronized
  5. SymbolInfoDouble
  6. SymbolInfoInteger
  7. SymbolInfoString
  8. SymbolInfoTick
  9. SymbolInfoSessionQuote
  10. SymbolInfoSessionTrade
  11. MarketBookAdd
  12. MarketBookGet
- Bir sembolün/periyodun zaman-serilerinin alınması için şu fonksiyonlarla yapılan istek:
  1. CopyBuffer
  2. CopyRates
  3. CopyTime
  4. CopyOpen
  5. CopyHigh
  6. CopyLow
  7. CopyClose
  8. CopyTickVolume
  9. CopyRealVolume
  10. CopySpread

"Başka" bir sembol ilk defa çağrıldığında, sınama süreci durdurulur ve sembolün/periyodun geçmişi terminalden sınama temsilcisine aktarılır. Bu sembol için gereken tik dizisi de aynı zamanda oluşturulur.

Seçilen tik oluşturma moduna göre, her bir sembol için ayrı tik tik dizileri oluşturulur. Ayrıca, SymbolSelect() fonksiyonunu OnInit() işleyicisi içinde kullanarak, sembolün geçmişini açık yolla isteyebilirsiniz - geçmiş verinin yüklenmesi işlemi, Uzman Danışmanın sınanmasına başlamadan hemen önce gerçekleştirilir.

Bu şekilde, MetaTrader 5 müşteri terminali içinde çok-dövizli bir sınama gerçekleştirmek için ayrı bir çabaya ihtiyaç olmaz. Sadece, uygun sembollerin çizelgelerini müşteri terminalinde açmanız yeterlidir. İstenen veriyi içeren geçmiş veri, istenen tüm semboller için, alım-satım sunucusundan otomatik olarak indirilecektir.

Strateji Sınama Aracında Zamanın Simülasyonu

Sınama sırasında yerel zaman TimeLocal() her zaman sunucu zamanına eşittir TimeTradeServer(). Dolayısıyla, sunucu zamanı daima GMT'ye karşılık gelen zamana eşit olacaktır - TimeGMT(). Bu sebeple, tüm bu fonksiyonlar sınama sırasında aynı zamanı gösterecektir.

Strateji Sınama Aracında GMT zamanı, yerel zaman ve sunucu zamanı arasında fark olmaması, sunucu bağlantısının kurulmaması sebebiyle kasıtlı olarak yapılmıştır. Sınama sonuçları, bağlantı olup olmadığına bakılmaksızın her zaman aynı olmalıdır. Sunucu zamanı hakkındaki bilgi yerel olarak depolanmaz ve sunucudan alınır.

Sınama içinde Grafiksel Nesneler

Sınama/optimizasyon sırasında, grafiksel nesneler çizilmez. Bu nedenle, sınama/optimizasyon sırasında oluşturulmuş grafiksel nesnelerin özelliklerine başvurulduğunda, Uzman Danışman sıfır değerlerini alacaktır.

> Bu kısıtlama görsel sınama modunda uygulanmaz.

Strateji Sınayıcıda OnTimer() Fonksiyonu

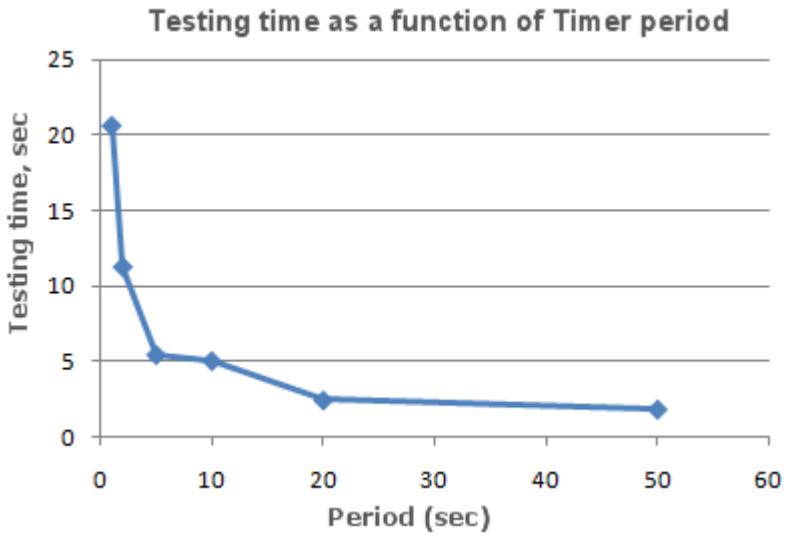
MQL5 zamanlayıcı olaylarının işlenmesine izin verir. OnTimer() işleyicisinin çağrısı, sınama modundan bağımsızdır. "Sadece açılış fiyatları" modunda H4 periyodu için bir sınama gerçekleştirildiğini ve uzman danışmanın saniyelik çağrılar yapan bir zamanlayıcıya sahip olduğunu düşünelim; bu durumda, H4 çubuğunun her açılış anında OnTick() işleyicisi bir defa, OnTimer() işleyicisi ise 14400 defa (3600 saniye * 4 saat) çağrılacaktır. Bu miktar Uzman Danışmanın sınama süresi ve çalışma mantığına bağlı olarak aratacaktır.

Sınama süresinin zamanlayıcı frekansına olan bağımlılığını ölçmek için, alım-satım işlemi içermeyen basit bir Uzman Danışman oluşturduk.

```
//--- giriş parametreleri
input int      timer=1;              // timer zamanlayıcı değeri, saniye
input bool     timer_switch_on=true; // zamanlayıcı devrede
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- timer_switch_on==true ise zamanlayıcıyı çalıştır
   if(timer_switch_on)
     {
      EventSetTimer(timer);
     }
```

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- zamanlayıcıyı durdur
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| Timer function                                                   |
//+------------------------------------------------------------------+
void OnTimer()
  {
//---
// herhangi bir eylem yok, işleyicinin gövdesi boş
  }
//+------------------------------------------------------------------+
```

Sınama süresi ölçümlerinde, timer parametresine bağlı olarak farklı değerler alınmıştır(Timer olayının periyodikliği). Elde edilen veriye göre, sınama süresini Timer periyodunun bir fonksiyonu olarak çizdik.



Grafikten de görebileceğiniz gibi, EventSetTimer(Timer) fonksiyonunun başlatılması sırasında timer parametresi ne kadar küçükse, OnTimer() işleyicisinin çağrıları arasındaki periyot o kadar küçük ve sınama süresi T o kadar fazladır.

Strateji Sınayıcıda Sleep() Fonksiyonu

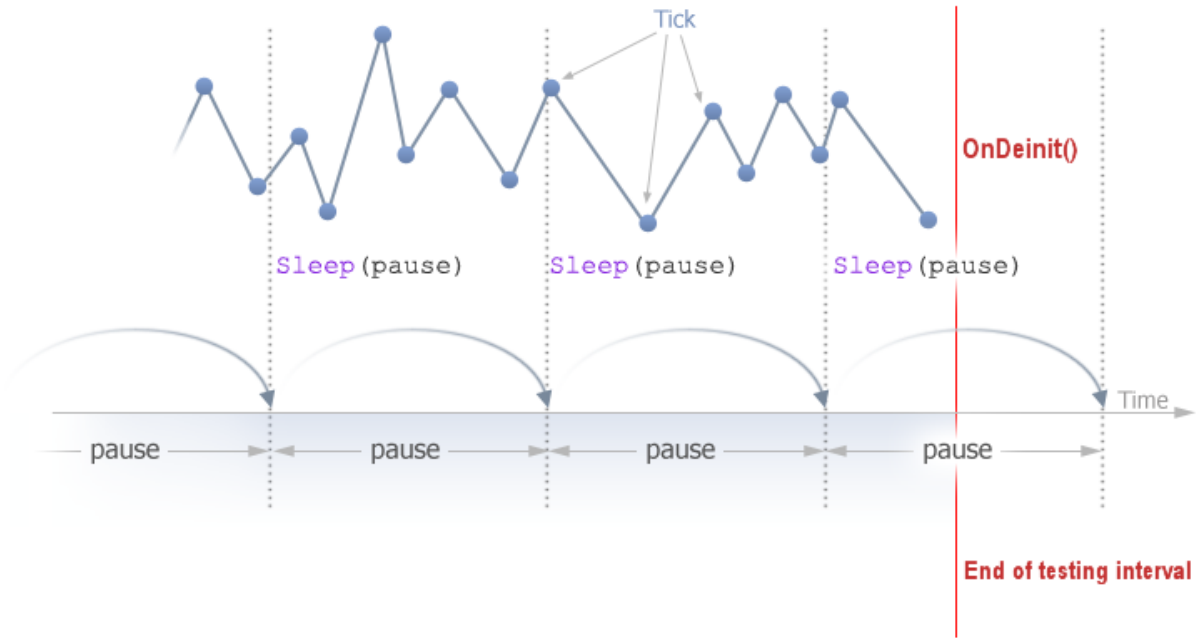
Grafik üzerinde çalışırken Sleep() fonksiyonu, Uzman Danışmana veya betiğe, mql5-programının çalışmasını bir süreliğine durdurma imkanı sağlar. Bu, o an hazır olmayan bir veri istediğinizde, veri

hazır olana kadar beklemeniz için kullanışlıdır. Sleep() fonksiyonunun detaylı kullanımına dair bir örnek Veri Erişiminin Düzenlenmesi bölümünde bulunabilir.

Sınama süreci, Sleep() çağrısıyla duraklamaz. Sleep() fonksiyonunu çağırdığınızda, oluşturulan tikler belirtilen gecikme içinde yürütülmeye devam eder - ki bu, bekleyen emirlerin ve durdurma emirlerinin tetiklenmesiyle sonuçlanabilir. Sleep() çağrısının ardından, Strateji Sınama aracında simüle edilen zaman, Sleep fonksiyonuyla belirtilen süre kadar artırılır.

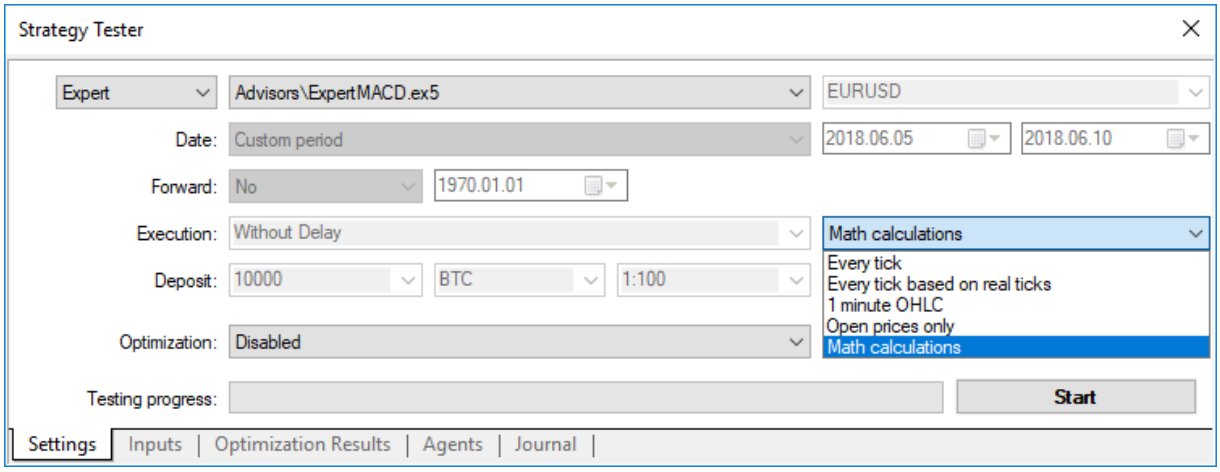
Sleep() fonksiyonunun çalıştırılması sırasında, Strateji Sınama Aracındaki mevcut zaman sınama periyodunun ötesine geçiyorsa; "Sınama sırasında sonsuz Sleep döngüsü tespit edildi" şeklinde bir hata mesajı alırsınız. Bu hata mesajını almışsanız, test sonuçları reddedilmez, tüm hesaplamalar tam hacmiyle (işlemlerin sayısı) gerçekleştirilir ve test sonuçları terminale aktarılır.

Sınama süresi, sınama aralığının boyutunu aşacağından, Sleep() fonksiyonu OnDeinit() içerisinde çalışmayacaktır, .



Matematiksel Hesaplamalardaki Optimizasyon Problemleri için Strateji Sınama aracının Kullanılması

MetaTrader 5 terminalindeki sınayıcı, sadece alım-satım stratejileri için değil, aynı zamanda matematiksel hesaplamalar için de kullanılabilir. Bunu kullanmak için "Matematiksel hesaplamalar" modunu seçmeniz gerekmektedir:



Bu durumda sadece üç fonksiyon çağrılacaktır: OnInit(), OnTester(), OnDeinit(). "Matematiksel hesaplamalar" modunda Strateji Sınama Aracı herhangi bir tik oluşturmayacak ve geçmiş veri yüklemeyecektir.

Strateji sınayıcı, başlangıç tarihini bitiş tarihinden büyük yazmış olsanız bile "Matematiksel hesaplamalar" modunda çalışır.

Sınama aracını matematiksel problemleri çözmek için kullanırken, tikler oluşturulmayacak ve geçmiş veri yüklenmeyecektir.

MetaTrader 5 Strateji Sınama Aracında çözmek için tipik bir matematiksel problem - çok değişkenli bir fonksiyonun ekstremumunun aranması.

Bunun için şu koşular sağlanmalıdır:

- Fonksiyon değerinin hesaplama işlemleri OnTester() fonksiyonu içine yerleştirilmelidir;
- Fonksiyonun parametreleri, Uzman Danışmanın giriş parametreleri olarak belirlenmelidir;

Uzman Danışmanı derleyin ve "Strateji Sınama Aracı" penceresini açın. "Giriş parametreleri" sekmesinde, istenen giriş parametrelerini seçin ve başlangıç, durdurma ve adım değerlerini belirleyerek, her bir fonksiyon parametresinin değerini tanımlayın.

Optimizasyon tipini seçin - "Yavaş, tam algoritma" (parametre uzayının tamamının aranması) veya "Hızlı, genetik tabanlı algoritma". Fonksiyonun ekstremum değerini basitçe aramak için, hızlı optimizasyonun seçilmesi daha iyi olacaktır; ama bütün parametre kümesi için arama gerçekleştirmek istiyorsanız, o zaman en iyisi yavaş optimizasyondur.

"Matematiksel hesaplamalar" modunu seçin ve "Başlat" düğmesine basarak optimizasyon prosedürünü çalıştırın. Optimizasyon sırasında Strateji Sınayıcının, OnTester fonksiyonunun maksimum değerini arayacağını not edin. Eğer yerel minimum değerini bulmak istiyorsanız, hesaplanan fonksiyonun değerini OnTester fonksiyonu içinde ters çevirin:

```
return(1/fonksiyon_değeri);
```

Fonksiyon değerinin sıfıra eşit olmadığı kontrol edilmelidir, aksi durumda sıfıra bölmeden kaynaklanan bir kritik hata alırız.

Bununla birlikte bir yol daha bulunmaktadır, bu yöntem daha uygundur ve optimizasyon sonuçlarında bozulmaya yol açmaz, bu makalenin okuyucuları tarafından önerilmiştir:

```
return(-fonksiyon_değeri);
```

Bu seçenekte fonksiyon değerinin sıfıra eşit olup olmadığının kontrol edilmesi gerekmez, 3-boyutlu bir temsilde optimizasyon yüzeyi aynı şekle sahiptir. Buradaki tek fark bunun orjinaline göre bir yansıma olmasıdır.

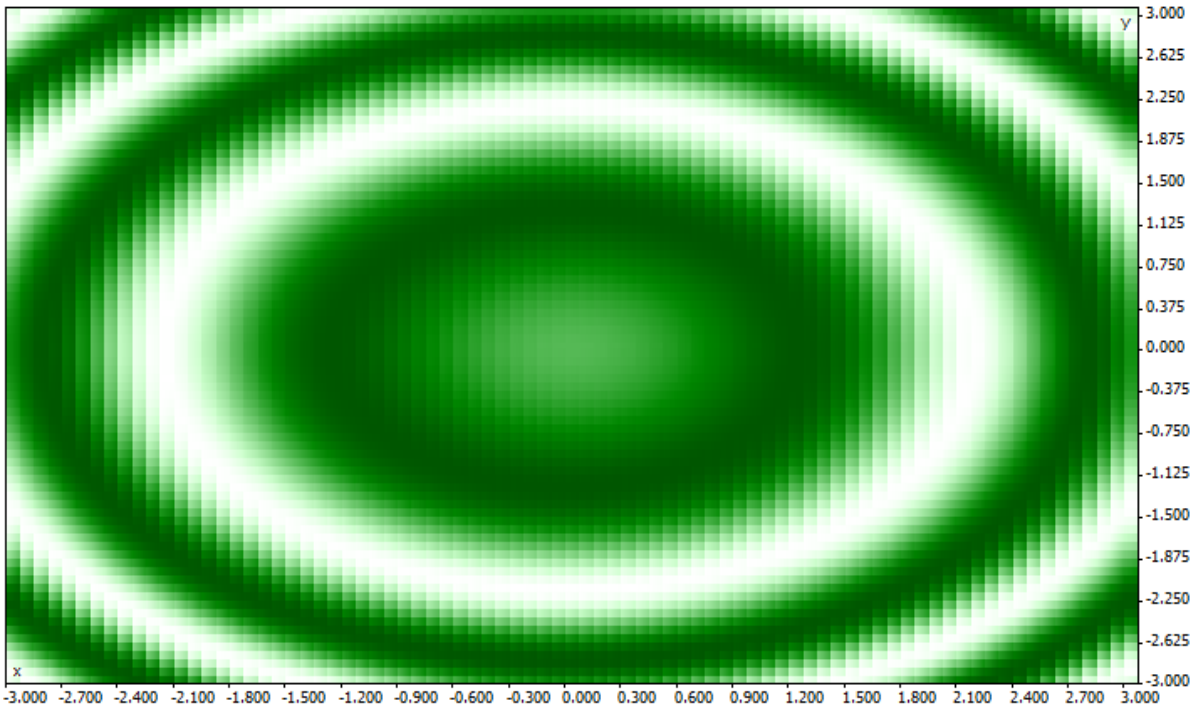
Örnek olarak sink() fonksiyonunu kullanalım:

$$sink(x,y) = \sin(x^2 + y^2)$$

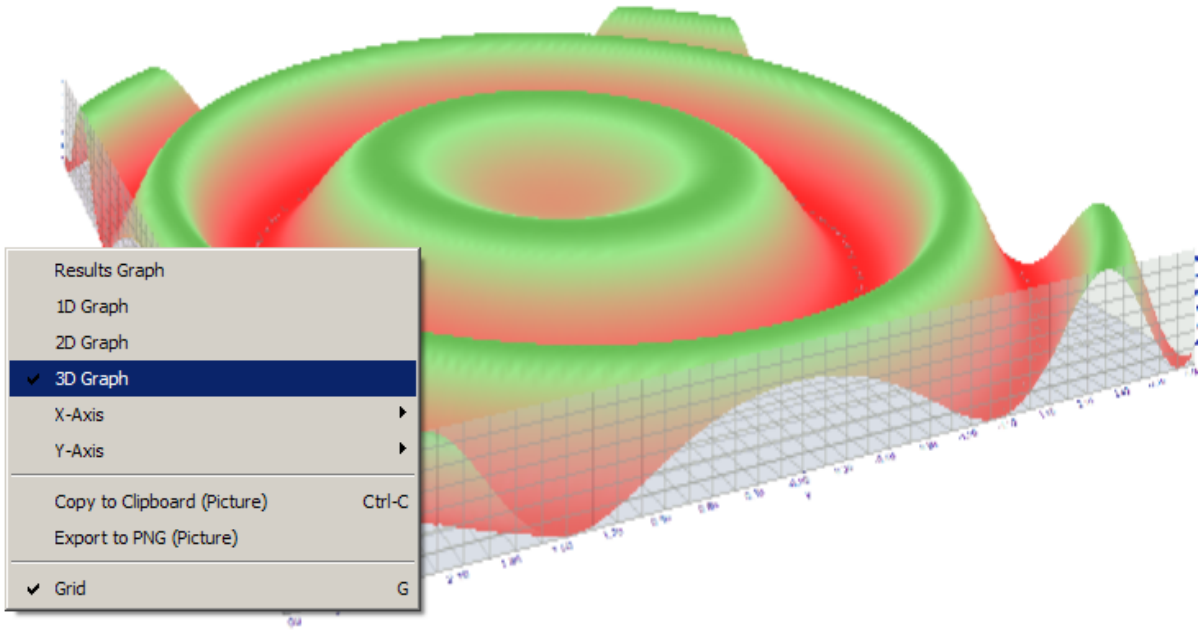
Bu fonksiyonun ekstremumunu bulmak için gereken Uzman Danışman kodu OnTester() fonksiyonunun içine yerleştirilir:

```
//+------------------------------------------------------------------+
//|                                                         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"
//--- giriş parametreleri
input double   x=-3.0; // başlangıç=-3, adım=0.05, durdurma=3
input double   y=-3.0; // başlangıç=-3, adım=0.05, durduma=3
//+------------------------------------------------------------------+
//| Tester function                                                  |
//+------------------------------------------------------------------+
double OnTester()
  {
//---
   double sink=MathSin(x*x+y*y);
//---
   return(sink);
  }
//+------------------------------------------------------------------+
```

Bir optimizasyon gerçekleştirin ve optimizasyon sonuçlarını 2-boyutlu grafik şeklinde alın.



Verilen parametre değerleri (x, y) ne kadar iyiyse, renk de o kadar canlı olacaktır. sink() formülünden umulduğu gibi, fonksiyonun değerleri (0,0) merkezinin etrafında konsantrik daireler şeklindedir. sink() fonksiyonunun tek bir global maksimuma sahip olmadığı, 3-boyutlu grafikten görülebilir:



"Sadece Açılış fiyatları" modunda Çubukların Senkronizasyonu

MetaTrader 5 müşteri terminalindeki sınayıcı, "çok-dövizli" Uzman Danışmanları kontrol etmemizi sağlar. Çok-dövizli bir Uzman Danışman, iki veya daha çok sembol üzerinden alım-satım gerçekleştiren bir Uzman Danışmandır.

Birden çok sembol üzerinden alı-satım yapan stratejilerin sınanması, sınayıcıyaa birkaç ek teknik gereksinim yükler:

- Bu semboller için tiklerin oluşturulması;
- Gösterge değerlerinin bu semboller için hesaplanması;
- Marjin (teminat) gereksinimlerinin bu sembol için hesaplanması;
- Oluşturulan tik dizilerinin tüm alım-satım sembolleri için senkronize edilmesi.

Strateji sınayıcı, her bir enstrüman için seçilen alım-satım moduna göre bir tik dizisini oluşturur ve yürütür. Aynı zamanda, her bir sembol için yeni bir çubuk diğer sembollerden bağımsız olarak açılır. Yani, çok-dövizli Uzman Danışmanların sınanması sırasında, bir enstrüman için yeni çubuğun çoktan açıldığı, diğeri içinse henüz açılmadığı bir durumla karşılaşılabilir (bu durum sıklıkla yaşanır). Böylece, sınama içindeki her şey gerçekte olduğu gibi olur.

Sınama aracında kullanılan bu özgün simülasyon yöntemi, "Her tik" ve "1 dakikalık OHLC" modları seçildiği sürece herhangi bir sorun ortaya çıkarmaz. Bu modlarda, farklı sembollerdeki çubukların senkronizasyonunun gerçekleşmesini bekleyebilmek amacıyla, her bir çubuk için yeterli sayıda tik üretilir. Peki alım-satım enstrümanlarının senkronizasyonu bu kadar gerekliyse, çok-dövizli bir Uzman Danışmanı, "Sadece açılış fiyatları" modunda nasıl sınayabiliriz? Bu modda Uzman Danışman sadece açılış zamanına denk gelen tek bir tik üzerinde çağrılır.

Bunu bir örnek üzerinde göstereceğiz: EURUSD üzerinde bir uzmanı test ediyor olalım ve H1 çizelgesinde yeni saatin mumu açılmış olsun. Bu olayı kolayca anlayabiliriz - "Sadece açılış fiyatları" modunda NewTick olayı, sınama periyodunun üzerindeki çubuk açılış zamanına denk gelecektir. Ama yeni çubuğun Uzman Danışmanda kullanılan USDJPY sembolü üzerinde de açılacağı kesin değildir.

Normal koşullar altında, OnTick() fonksiyonunun işlemini tamamlamak ve bir sonraki tik ile birlikte USDJPY sembolü üzerindeki yeni çubuğu kontrol etmek yeterlidir. Ama "Sadece açılış fiyatları" modunda bir sınama gerçekleştirirken başka bir tik olmayacaktır; bu yüzden bu modun çok-dövizli Uzman Danışmanların sınanması için uygun olmadığı düşünülebilir. Ama öyle değildir - MetaTrader 5 içindeki sınayıcının gerçek hayattaki gibi davrandığını unutmamanız gerekir. Sleep() fonksiyonunu kullanarak, diğer semboller için yeni çubuklar açılana kadar bekleyebilirsiniz!

Synchronize_Bars_Use_Sleep.mq5 Uzman Danışmanının kodu, "Sadece açılış fiyatları" modunda çubukların senkronizasyonunun bir örneğini göstermektedir:

```
//+------------------------------------------------------------------+
//|                                   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"
//--- giriş parametreleri
input string   other_symbol="USDJPY";
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- sembolü kontrol et
   if(_Symbol==other_symbol)
     {
```

```
      PrintFormat("Diğer sembolü giriş parametrelerinde belirtmeli veya Strateji Sınar
      //--- sınama zorla durduruldu
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- son çubuğun zamanını saklaması için bir static değişken
   static datetime last_bar_time=0;
//--- senkronizasyon bayrağı
   static bool synchonized=false;
//--- Eğer statik değişken başlatılmamışsa
   if(last_bar_time==0)
     {
      //--- ilk çağrı, çubuk zamanını sakla ve çık
      last_bar_time=(datetime)SeriesInfoInteger(_Symbol,Period(),SERIES_LASTBAR_DATE);
      PrintFormat("last_bar_time değişkeni %s değeri ile başlatıldı",TimeToString(last
     }
//--- sembolün son çubuğunun açılış zamanını al
   datetime curr_time=(datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_LASTBAR_DAT
//--- zamanlar eşit değilse
   if(curr_time!=last_bar_time)
     {
      //--- çubuk açılış zamanını statik değişkene kaydet
      last_bar_time=curr_time;
      //--- senkronize değil
      synchonized=false;
      //--- mesajı çıktıla
      PrintFormat("%s sembolü üzerinde %s zamanında yenibir çubuk alındı",_Symbol,Time
     }
//--- diğer sembol çubuğunun açılış zamanı
   datetime other_time;
//--- diğer sembol çubuğunun açılış zamanı curr_time değişkenine eşit olana kadar dön
   while(!(curr_time==(other_time=(datetime)SeriesInfoInteger(other_symbol,Period(),SE
     {
      PrintFormat("5 saniye bekleniyor..");
      //--- beş saniye bekle ve SeriesInfoInteger(other_symbol,Period(),SERIES_LASTBAR
      Sleep(5000);
     }
//--- çubuklar senkronize edildi
   synchonized=true;
   PrintFormat("%s sembolünün çubuk açılış zamanı: %s",_Symbol,TimeToString(last_bar_t
   PrintFormat("%s sembolünün çubuk açılış zamanı: %s",other_symbol,TimeToString(other
//--- TimeCurrent() kullanışsız, TimeTradeServer() çağrısını kullan
```

```
    Print("Çubukların senkronizasyon zamanı ",TimeToString(TimeTradeServer(),TIME_SECON
  }
//+------------------------------------------------------------------+
```

Uzman Danışmanın son satırına dikkat edin - bu satırda senkrinazyonun gerçekleştirildiği mevcut zaman görüntülenmektedir:

```
    Print("Çubukların senkronizasyon zamanı ",TimeToString(TimeTradeServer(),TIME_SECON
```

Mevcut zamanı görüntülemek için TimeCurrent() fonksiyonunun yerine TimeTradeServer() fonksiyonunu kullandık. TimeCurrent() fonksiyonu, Sleep() çağrısından sonra değişmeyen 'son tik zamanına' dönüş yapar. Bu Uzman Danışmanı "Sadece açılış fiyatları" modunda çalıştırdığınızda, çubukların senkronizasyonuna dair bir mesaj görürsünüz.

Core 1	2010.12.01 20:00:05	The bars are synchronized at 2010.12.01 20:00:05
Core 1	2010.12.01 20:00:05	Open bar time of the chart symbol USDJPY: 2010.12.01 20:00
Core 1	2010.12.01 20:00:05	A new bar has appeared on symbol EURUSD: 2010.12.01 20:00
Core 1	2010.12.01 20:00:00	Waiting 5 seconds..
Core 1	2010.12.01 20:00:05	A new bar has appeared on symbol EURUSD: 2010.12.01 20:00
Core 1	2010.12.01 16:00:05	The bars are synchronized at 2010.12.01 16:00:05

Son tik zamanı yerine mevcut sunucu zamanını almanız gerekiyorsa, TimeCurrent() yerine TimeTradeServer() fonksiyonunu kullanın.

Çubukların senkronizasyonu için bir yol daha mevcuttur - zamanlayıcı kullanımı. Bu tip bir Uzman Danışman örneği (Synchronize_Bars_Use_OnTimer.mq5) bu makaleye eklenmiştir.

Sınayıcı içinde IndicatorRelease() Fonksiyonu

Tekil bir sınama tamamladıktan sonra, otomatik olarak sembolün bir çizelgesi açılır. Bu çizelgede, tamamlanan işlemler ve Uzman Danışmanda kullanılan göstergeler yer alır. Bu, giriş-çıkış noktalarını görsel olarak analiz etmenize ve bunları gösterge değerleriyle karşılaştırmanıza olanak sağlar.

> **Not:** sınama işleminden sonra açılan çizelgede görüntülenen göstergeler, sınamanın tamamlanmasından sonra hesaplanır. Bu göstergeler Uzman Danışman İçerisinde kullanılmış olsa bile.

Ama bazı durumlarda, programcı hangi göstergelerin alım-satım algoritmasına dahil olduğunu gizlemek isteyebilir. Örneğin, gizlenmiş kodlarla, çalıştırılabilir bir dosya şeklinde satılan veya kiralanan bir Uzman Danışman. Bu amaç için, IndicatorRelease() fonksiyonunun kullanımı uygundur.

Eğer terminal tarafından, directory/profiles/templates konumundan tester.tpl isimli bir şablon ayarlanırsa, bu şablon açılan çizelgeye uygulanacaktır. Bu şablonun yokluğu durumunda ise, varsayılan şablon uygulanacaktır. (default.tpl).

Aslında IndicatorRelease() fonksiyonu, kullanımı tamamlanan bir göstergenin hesaplama kısmını serbest bırakmak için tasarlanmıştır. Bunun amacı bellek ve CPU kaynaklarını korumaktır, çünkü her yeni tikten sonra gösterge hesaplanması için yeniden çağrılır. İkinci bir amaç ise, tekil bir sınamadan sonra göstergenin sınama çizelgesinde görüntülenmesini engellemektir.

Sınama işleminden sonra göstergenin çizelge üzerinde görüntülenmesini engellemek için, IndicatorRelease() fonksiyonunu, gösterge tanıtıcısı ile OnDeinit() işleyicisinin içinde çağırın. OnDeinit() fonksiyonu her zaman sınama tamamlandıktan sonra ve sınama çizelgesi gösterilmeden önce çalışır.

```
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   bool hidden=IndicatorRelease(handle_ind);
   if(hidden) Print("IndicatorRelease() başarıyla tamamlandı");
   else Print("IndicatorRelease()'false' dönüşü yaptı Hata kodu ",GetLastError());
  }
```

Sınama işleminden sonra göstergenin çizelge üzerinde görüntülenmesini engellemek için IndicatorRelease() fonksiyonunu OnDeinit() işleyicisinin içerisinde çağırın.

Sınama Aracında Olay İşleme

MetaTrader 5 sınayıcıda geçmiş veri ile sınamaya konu olması açısından OnTick() işleyicisinin Uzman Danışman içindeki varlığı zorunlu değildir. Bir Uzman Danışmanın aşağıdaki işleyicilerden en az birini içermesi yeterlidir:

- OnTick() - gelen yeni tik olayının işleyicisi;
- OnTrade() - Alım-satım olayının işleyicisi;
- OnTimer() - Zamanlayıcıdan gelen sinyal olaylarının işleyicisi;
- OnChartEvent() - Özel olaylar için bir işleyici.

Bir Uzman Danışmanı sınarken OnChartEvent() fonksiyonunu kullanarak özel olayları işleyebiliriz ama bu fonksiyon göstergelerde sınayıcı içinde çağrılamaz Gösterge OnChartEvent() içeriyorsa ve gösterge sınanan uzman Danışmanda kullanılıyor olsa bile, gösterge herhangi bir özel olay işlemeyecektir.

Sınama sırasında, göstergeler EventChartCustom() fonksiyonu ile özel olaylar oluşturabilirler ve Uzman Danışman bu olayı OnChartEvent() içerisinde işleyebilir.

Bu olaylara ilaveten, sınama sürecine ve optimizasyona dair özel olaylar da strateji sınayıcıda oluşturulurlar:

- Tester - bu olay, Uzman Danışmanın sınama işleminin tamamlanmasından sonra oluşturulur. Tester olayı, OnTester() fonksiyonu ile işlenir. Bu olay sadece Uzman Danışmanların sınanmasında kullanabilir ve aslen giriş parametrelerinin genetik optimizasyonu için kullanılan Custom max kriterinin değerini hesaplamak amacıyla tasarlanmıştır.
- TesterInit - bu olay, strateji sınayıcıdaki optimizasyon işleminin başlangıcında, ilk geçişin öncesinde oluşturulur. TesterInit olayı, OnTesterInit() fonksiyonu ile işlenir. Optimizasyonun başlangıcında, bu işleyiciye sahip bir uzman danışman, sınayıcıda belirtilen sembolün ve periyodun çizelgesini otomatik olarak, ayrı bir terminal penceresi üzerinde açar. Fonksiyon, optimizasyona geçilmeden önce, daha sonraki optimizasyon sonuçlarının işlenmesi aşaması için, Uzman Danışmanlarda kullanılır..
- TesterPass - bu olay, yeni bir veri çerçevesi alındığında oluşturulur. TesterPass olayı OnTesterPass() fonksiyonu ile işlenir. Bu işleyiciye sahip bir Uzman Danışman, sınama için belirlenen sembol/zaman

aralığı değerleri ile otomatik olarak ayrı bir terminal grafiğine yüklenir, optimizasyon sırasında bir çerçeve alındığında TesterPass olaylarını yakalar. Fonksiyon, optimizasyon sonuçlarının dinamik olarak tamamlanmasını beklemeksizin "anında" işlenmesi için kullanılır. Çerçeveler, OnTester() işleyicisindeki bir tekil geçişin bitmesinden sonra çağrılabilen FrameAdd() işleyicisi kullanılarak eklenirler.

- TesterDeinit - bu olay, strateji sınayıcıda Uzman Danışman optimizasyonunun bitmesinin ardından oluşturulur.. TesterDeinit olayı, OnTesterDeinit() fonksiyonu ile işlenir. TesterDeinit() işleyicisine sahip bir Uzman Danışman, optimizasyonun başlangıcında otomatik olarak bir grafiğe yüklenir ve tamamlanmasının ardından TesterDeinit olayını teslim alır. Fonksiyon, tüm optimizasyon sonuçlarının son işlemleri için kullanılır.

Sınama Temsilcileri

MetaTrader 5 müşteri terminalinde gerçekleştirilen sınamalar sınama temsilcileri kullanılarak uygulanır. Yerel temsilciler, otomatik olarak oluşturulup devreye sokulurlar. Yerel temsilcilerin sayısı bilgisayardaki çekirdek sayısına karşılık gelir.

Her sınama temsilcisi, müşteri terminalinden bağımsız olarak global değişkenlerin kendisine has kopyasına sahiptir. Burada terminal, görevleri yerel veya uzak temsilcilere dağıtan bir sevk merkezi gibidir. Temsilci Uzman Danışmanın sınanması görevini tamamladıktan sonra, verilen parametrelerle birlikte, sonuçları terminale verir. Tek bir sınama için tek bir temsilci kullanılır.

Temsilci, terminalden alınan geçmiş verileri enstrümanların isimlerine göre ayrı klasörlere kopyalar - EURUSD geçmişi, EURUSD isimli bir klasöre kaydedilir. Enstrümanların geçmiş verileri kaynaklarına göre de ayrılır. Geçmiş verilerin depolanma yapısı şu şekilde gözükür:

```
tester_catalog\Agent-IPaddress-Port\bases\kaynak_ismi\history\sembol_ismi
```

Örneğin, MetaQuotes-Demo sunucusundan alınan EURUSD geçmişi, tester_catalog\Agent-127.0.0.1-3000\bases\MetaQuotes-Demo\EURUSD klasörüne kaydedilebilir.

Yerel temsilci, sınamanın tamamlanmasının ardından bekleme moduna geçer ve yeni bir görev için 5 dakika boyunca bekler, böylece bir sonraki çağrıda zaman harcanmaz. Bekleme süresinin bitmesiyle Yerel temsilci kapanır ve CPU belleğinden kaldırılır.

Kullanıcı tarafından (İptal et düğmesi ile) veya müşteri terminalinin kapatılması sonucunda sınama sürecinin erken tamamlanması durumunda, Tüm temsilcler hemen işlemlerini durdurur ve bellekten kaldırılır.

Terminal ve Sınama Temsilcisi arasında Veri Değişimi

Bir sınama gerçekleştirdiğinizde, müşteri terminali temsilciye gönderilmek üzere bir dizi parametre bloğu hazırlar:

- Sınama için giriş parametreleri (simülasyon modu, test aralığı, enstrümanlar, optimizasyon kriteri, vb.)
- "Piyasa Gözlemi" içinden seçilen sembollerin listesi
- Sınanacak sembolün özellikleri (sözleşme büyüklüğü, StopLoss ve Takeprofit değerlerini ayarlamak için izin verilen teminatlar, vb.)
- Sınanacak Uzman Danışman ve onun giriş parametrelerinin değerleri
- Ek dosyalar hakkında bilgi (kütüphaneler, göstergeler, veri dosyaları - # property tester_ ...)

tester_indicator	string	"indicator_name.ex5" biçiminde bir özel göstergenin ismi. Sınama gerektiren özel göstergeler, iCustom() fonksiyonunun çağrısıyla otomatik olarak tanımlanırlar (karşılık gelen parametre, bir sabit dize ile ayarlanmışsa). Tüm diğer durumlar için (IndicatorCreate() fonksiyonunun kullanımı veya gösterge ismini ayarlayan parametrenin sabit-olmayan bir dize ile verilmesi) bu özellik istenir
tester_file	string	Sınayıcı için, uzantısı belirtilmiş şekilde, çift tırnak içinde (bir sabit dize olarak) dosya adı. Belirtilen dosya sınayıcıya aktarılır. Test edilecek girdi dosyaları - gerekli olanlar her zaman belirtilmelidir.
tester_library	string	Çift tırnak içinde, uzantılı olarak kütüphane adı. Bir kütüphane dll veya ex5 uzantılarına sahip olabilir. Sınama gerektiren kütüphaneler otomatik olarak tanımlanırlar. Ama, kütüphanelerden herhangi biri, bir özel gösterge tarafından kullanılıyorsa bu özellik gereklidir

Temsilciye gönderilecek her parametre bloğu için, MD5-hash biçiminde bir dijital imza oluşturulur. MD5-hash, her küme için benzersizdir; büyüklüğü hesaplandığı bilgiden kat kat daha küçüktür.

Temsilci, blokların hash değerlerini alır ve onları zaten elinde olanlarla karşılaştırır. Verilen parametrenin parmak-izi temsilcinin elinde yoksa veya alınan hash değeri temsilcide mevcut değilse, temsilci bu parametre bloğunu kabul eder. Bu, terminal ile sınama temsilcisi arasındaki trafiği azaltır.

Sınama işlemi bittikten sonra temsilci tüm sonuçları terminale döndürür. Bu sonuçlar, "Test Sonuçları" ve "Optimizasyon Sonuçları" isimli sekmelerde gösterilir: alınan kar, işlem sayısı, Sharpe katsayısı, OnTester() fonksiyonunun sonucu, vb.

Optimizasyon sırasında terminal sınama görevlerini küçük paketler halinde temsilcilere dağıtır, her paket birkaç görev içerir (her görev, bir giriş parametreleri kümesiyle tekil bir sınma anlamına gelir). Bu, terminal ve temsilci arasındaki değişim süresini azaltır.

Temsilciler, güvenlik nedeniyle, terminalden alınan EX5-dosyalarını hiçbir zaman sabit diske kaydetmez. Bu yüzden, temsilcinin çalıştığı bilgisayar gönderilen veriyi kullanamaz. Tüm diğer dosyalar - DLL dosyaları da dahil olmak üzere - güvenlik-ortamı (sandbox) içinde tutulur. DLL kullanan Uzman Danışmanları uzak temsilcilerde sınayamazsınız.

Sınama sonuçları, gerektiğinde hızlı bir şekilde ulaşabilmek amacıyla, terminal tarafından özel bir ön-bellekte toplanır (sonuç ön-belleği). Terminal her parametre kümesi için bir önceki çalışmadan gelen mevcut sonuçları, yeniden işlenmesini önlemek amacıyla sonuç ön-belleği içinde kontrol eder. Söz konusu parametre kümesine dair bir sonuç bulunamamışsa, görev, işlenmesi için temsilciye verilir.

Terminal ve sınama temsilcisi arasındaki tüm trafik şifrelenmiştir.

Tikler ağ üzerinden gönderilmez; sınama temsilcilerinde oluşturulurlar.

Tüm Müşteri Terminallerinin Ortak Klasörünün Kullanılması

Tüm sınama temsilcileri, birbirlerinden ve terminalden izole edilmiştir: her temsilci, günlüğünün kaydedildiği kendine has bir klasöre sahiptir. Bunun yanında, tüm dosya işlemleri

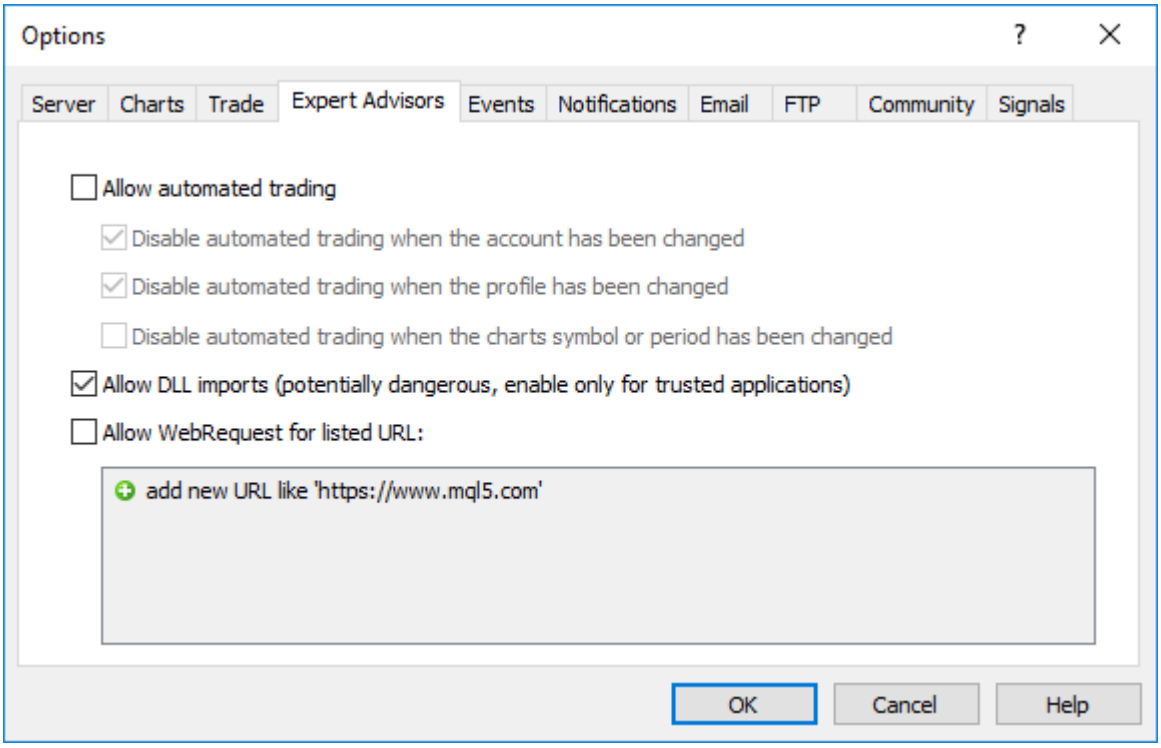
temsilci_ismi/MQL5/Files klasöründe gerçekleşir. Ama, tüm müşteri terminallerinin ortak klasörünü kullanarak, yerel temsilciler ve terminal arasında bir etkileşim sağlayabiliriz. Bunun için, dosya açılışı kısmında FILE_COMMON bayrağını belirtmeniz gerekir:

```
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- tüm müşteri terminalleri için paylaşılan klasör
   common_folder=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- klasörün ismini çıktıla
   PrintFormat("Dosyayı, tüm müşteri terminallerinin paylaşılan klasöründe aç %s", cor
//--- dosyayı paylaşılan klasörde aç (indicated by FILE_COMMON flag)
   handle=FileOpen(filename,FILE_WRITE|FILE_READ|FILE_COMMON);
  ... daha sonraki eylemler
//---
   return(INIT_SUCCEEDED);
  }
```

DLL Dosyalarının Kullanımı

Optimizasyonu hızlandırmak amacıyla sadece yerel temsilcileri değil, aynı zamanda uzak temsilcileri de kullanabiliriz. Bu durumda, bazı kısıtlamalar bulunmaktadır. Öncelikle, Print() fonksiyonunun çalıştırılmasından doğan sonuçları ve pozisyon açıp kapama mesajlarını günlüklerinde göstermezler. Hatalı yazılmış Uzman Danışmanların, uzak temsilcilerin bulunduğu bilgisayarı mesajlarla çöplüğe çevirmemesi için, günlük içinde sadece gereken bilgiler görüntülenir.

İkinci bir kısıtlama - Uzman Danışmanların sınanması sırasında DLL kullanımına izin verilmez. Güvenlik nedeniyle, uzak temsilciler üzerinde DLL çağrılarına kesinlikle izin verilmez. Yerel temsilcilerde ise sınanmış Uzman Danışmanın DLL kullanmasına sadece uygun şekilde izin verilir - "DLL kullanımına izin ver"



Not: DLL kullanımı gerektiren bir 3.parti Uzman Danışman (betik, gösterge) kullanırken, terminal ayarlarından bu özelliğe izin verdiğinizde aldığınız riski iyi anlamalısınız. Bu risk Uzman Danışmanın ne için kullanılacağıyla alakalı değildir.

Öntanımlı Değişkenler

Çalıştırılabilir her MQL5 programı için birtakım ön-tanımlı değişkenler sağlanmıştır. Bunlar, bir programın (Uzman Danışman, script veya özel gösterge) başlatılmasıyla birlikte, mevcut fiyat çizelgesinin durumuna tepki verirler.

Ön-tanımlı değişkenlerin değerleri MQL5 programı başlatılmadan önce müşteri terminali tarafından ayarlanır. Bu değişkenler sabittirler ve MQL5 programı içinde değiştirilemezler. İstisna olarak, sadece _LastError özel değişkeni bulunmaktadır, bu değişken ResetLastError fonksiyonu ile sıfırlanabilir.

Değişken	Değer
_AppliedTo	_AppliedTo değişkeni, gösterge hesaplaması için kullanılan veri türünü bulmayı sağlar
_Digits	Ondalık hanelerin sayısı
_Point	Karşıt döviz birimi cinsinden mevcut sembolün puan büyüklüğü
_LastError	Son hata kodu
_Period	Mevcut çizelgenin zaman-aralığı
_RandomSeed	Pseudo-rassal tamsayı üretecinin mevcut durumu
_StopFlag	Program durdurma bayrağı
_Symbol	Mevcut çizelgenin sembol ismi
_UninitReason	Sonlandırma sebebi kodu
_IsX64	_IsX64 değişkeni, MQL5 uygulamasının çalıştığı terminalin bit sürümünü bulmayı sağlar

Ön-tanımlı değişkenler kütüphane içinde tanımlanamazlar. Kütüphaneler, çağrıldıkları programda tanımlanan değişkenleri kullanırlar.

int _AppliedTo

_AppliedTo değişkeni, gösterge hesaplaması için kullanılan veri türünü bulmayı sağlar:

Data türü	Anlam	Gösterge hesaplaması için kullanılan verilerin açıklaması.
—	0	Gösterge, ikinci OnCalculate() çağrı formunu kullanır - hesaplama verileri belirli bir arabellek veya veri dizisi tarafından belirtilmez
Kapanış	1	Kapanış fiyatı
Açılış	2	Açılış fiyatı
Yüksek	3	Yüksek fiyat
Düşük	4	Düşük fiyat
Medyan Fiyat (HL/2)	5	Medyan fiyat = (Yüksek+Düşük)/2
Tipik Fiyat (HLC/3)	6	Tipik fiyat = (Yüksek+Düşük+Kapanış)/3
Ağırlıklı Fiyat (HLCC/4)	7	Ağırlıklı fiyat = (Açılış+Yüksek+Düşük+Kapanış)/4
Önceki indikatörün datası	8	Bu göstergeden önce grafikte başlatılan göstergenin verileri
İlk indikatörün datası	9	İlk olarak grafikte başlatılan göstergenin verileri
Gösterge kolu	10+	Gösterge kolunu kullanarak iCustom() işlevine iletilen göstergenin verileri. _AppliedTo değeri indikatör kounu içerir

Örnek:

```
//+------------------------------------------------------------------+
//| Özel gösterge başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge arabellekleri eşlemesi
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
// Gösterge hesaplaması için kullanılan veri tipinin elde edilmesi
   Print("_AppliedTo=",_AppliedTo);
   Print(getIndicatorDataDescription(_AppliedTo));
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

```
//| Gösterge hesaplaması için kullanılan verilerin açıklaması        |
//+------------------------------------------------------------------+
string getIndicatorDataDescription(int data_id)
  {
   string descr="";
   switch(data_id)
     {
      case(0):descr="It's first type of OnCalculate() - no data buffer";
         break;
      case(1):descr="Indicator calculates on Close price";
         break;
      case(2):descr="Indicator calculates on Open price";
         break;
      case(3):descr="Indicator calculates on High price";
         break;
      case(4):descr="Indicator calculates on Low price";
         break;
      case(5):descr="Indicator calculates on Median Price (HL/2)";
         break;
      case(6):descr="Indicator calculates on Typical Price (HLC/3)";
         break;
      case(7):descr="Indicator calculates on Weighted Price (HLCC/4)";
         break;
      case(8):descr="Indicator calculates Previous Indicator's data";
         break;
      case(9):descr="Indicator calculates on First Indicator's data";
         break;
      default: descr="Indicator calculates on data of indicator with handle="+string(c
         break;
     }
//---
   return descr;
  }
```

Ayrıca bakın

ENUM_APPLIED_PRICE

int _Digits

_Digits t, mevcut çizelge sembolünün fiyat değeri çözünürlüğünü belirleyen ondalık basamak sayısını depolar.

Bunun yerine, Digits() fonksiyonu da kullanılabilir.

double _Point

_Point değişkeni, karşıt döviz birimi cinsinden mevcut sembolün puan büyüklüğünü içerir.

Bunun yerine Point() fonksiyonunu da kullanabilirsiniz.

int _LastError

_LastError değişkeni, MQL5 programının çalıştırılması sırasında oluşturulan son hata kodunu içerir. Değeri, ResetLastError() fonksiyonu ile sıfırlanabilir.

Bununla birlikte, GetLastError() fonksiyonu da son hata kodunu almak için kullanılabilir.

ENUM_TIMEFRAMES _Period

_Period değişkeni mevcut çizelgenin zaman aralığını içerir.

Bunun yerine Period() fonksiyonu da kullanılabilir.

Ayrıca Bakınız

PeriodSeconds, Çizelge zaman aralıkları, Zaman ve Tarih, Nesnelerin Görünürlüğü

_RandomSeed

Pseudo-rassal tamsayıların üretilmesi sırasında, mevcut durumu içeren değişkendir. _RandomSeed değeri, MathRand() fonksiyonu çağrıldığında değişir. İstenen başlangıç koşulunu ayarlamak için MathSrand() fonksiyonunu kullanın.

MathRand() fonksiyonu ile elde edilen ***x*** rassal sayısı, her bir çağrıda şu yöntem kullanılarak hesaplanır:

```
x=_RandomSeed*214013+2531011;
_RandomSeed=x;
x=(x>>16)&0x7FFF;
```

Ayrıca Bakınız

MathRand(), MathSrand(), Tam-sayı tipleri

bool _StopFlag

_StopFlag değişkeni, MQL5 programının durdurma bayrağını içerir. Müşteri terminali programı durdurmak istediğinde, _StopFlag değişkenini 'true' değeri ile ayarlar.

_StopFlag değişkeninin durumunu öğrenmek için IsStopped() fonksiyonunu da kullanabilirsiniz.

string _Symbol

_Symbol değişkeni mevcut çizelgenin sembolünü içerir.

Bunun yerine Symbol() fonksiyonunu da kullanabilirsiniz.

int _UninitReason

_UninitReason değişkeni, programın sonlandırma sebebi kodunu içerir.

Bu kod genellikle UninitializeReason() fonksiyonu ile elde edilir.

int _IsX64

_IsX64 değişkeni, MQL5 uygulamasının çalıştığı terminalin bit sürümünü bulmayı sağlar: _IsX64=0 32-bit terminal için ve _IsX64!=0 64-bit terminal için.

Ayrıca, fonksiyon TerminalInfoInteger(TERMINAL_X64) kullanılabilir.

Örnek:

```
// Hangi programın çalıştığını anlamak için terminali kontrol etme
   Print("_IsX64=",_IsX64);
   if(_IsX64)
      Print("Program ",__FILE__," is running in the 64-bit terminal");
   else
      Print("Program ",__FILE__," is running in the 32-bit terminal");
   Print("TerminalInfoInteger(TERMINAL_X64)=",TerminalInfoInteger(TERMINAL_X64));
```

Ayrıca bakın

MQLInfoInteger, İçe aktarma fonksiyonları (#import)

Yaygın Fonksiyonlar

Hiçbir özelleştirilmiş grupta yer almayan genel amaçlı fonksiyonlar burada listelenmiştir.

Fonksiyon	Eylem
Alert	Ayrı bir pencerede bir mesaj gösterir
CheckPointer	Nesne işaretçisinin tipine dönüş yapar
Comment	Çizelgenin sol üst köşesine bir yorumu çıktılar
CryptEncode	Transforms the data from array with the specified method
CryptDecode	Performs the inverse transformation of the data from array
DebugBreak	Hata ayıklamadaki program kırılma noktası
ExpertRemove	Uzman Danışmanı durdurur ve çizelgeden kaldırır
GetPointer	Nesne işaretçisine dönüş yapar
GetTickCount	Sistemin başlatılmasından bu yana geçen milisaniyelerin sayısına dönüş yapar
GetTickCount64	Sistemin başlatılmasından bu yana geçen milisaniyelerin sayısına dönüş yapar
GetMicrosecondCount	Returns the number of microseconds that have elapsed since the start of MQL5-program
MessageBox	Bir mesaj kutusu oluşturur, görüntüler ve yönetir
PeriodSeconds	Periyottaki saniye sayısına dönüş yapar
PlaySound	Bir ses dosyasını oynatır
Print	Günlük içinde bir mesaj görüntüler
PrintFormat	Önceden ayarlanmış biçime uygun olarak, günlük içindeki sembol kümelerini biçimlendirip çıktılar
ResetLastError	Önceden belirlenmiş _LastError değerini sıfır yapar
ResourceCreate	Bir veri setinin temelinde, bir görüntü kaynağı oluşturur
ResourceFree	Dinamik olarak oluşturulmuş kaynağı siler, (bunun için ayrılmış belleği serbest bırakarak)
ResourceReadImage	ResourceCreate() fonksiyonuyla oluşturulan veya derleme sırasında EX5 dosyasına kaydedilen grafiksel kaynaktan veri okur
ResourceSave	Bir kaynağı belirtilen bir dosya içine kaydeder
SetUserError	Ön tanımlı _LastError değişkenini ERR_USER_ERROR_FIRST + user_error toplamına eşit olan değere ayarlar
SetReturnError	Operasyon tamamlanırken terminal işlemini geri döndüren kodu ayarlar

Fonksiyon	Eylem
Sleep	Mevcut Uzman Danışmanın veya betiğin çalışmasını, belirlenen bir aralık boyunca bekletir
TerminalClose	Terminale işlemi tamamlama komutu verir
TesterHideIndicators	Bir EA'da kullanılan displaying/hiding göstergelerinin modunu ayarlar
TesterHideIndicators	Bir EA'da kullanılan displaying/hiding göstergelerinin modunu ayarlar
TesterStatistics	Sınama sonuçlarına dayanarak hesaplanan belirli bir istatistiğin değerine dönüş yapar
TesterStop	Test etme sırasında program operasyonu tamamlanma komutunu verir.
TesterDeposit	Sınama sürecinde para yatırma işlemini taklit eden özel fonksiyon. Bazı para yönetim sistemlerinde kullanılabilir
TesterWithdrawal	Sınama sürecinde para çekme işlemini taklit eden özel bir fonksiyon
TranslateKey	Mevcut giriş dili ve kontrol anahtarının durumuna göre
ZeroMemory	Referansla geçirilen bir değeri sıfırlar. Değişken, yapıcılara sahip olan yapı ve sınıflar hariç her tipte olabilir

Alert

Ayrı bir pencerede bir mesaj gösterir.

```
void  Alert(
   argument,      // ilk değer
   ...            // diğer değerler
   );
```

Parametreler

argument

[in] Virgül ile ayrılmış herhangi bir değer. Bilgi çıktısını birkaç satıra bölmek için "\n" veya "\r\n" yeni satır karakterlerini kullanabilirsiniz. Parametrelerin sayısı 64'ü geçemez.

Dönüş değeri

Dönüş değeri yok.

Not

Diziler, Alert() fonksiyonuna geçirilemezler. Diziler elementsel şekilde çıktılanmalıdır. double tipli veriler, 8 ondalık ondalık haneye sahip çıktılar oluşturur, float tipi ise 5 ondalık haneyle gösterilir. Reel sayıları farklı çözünürlükle çıktılamak için, DoubleToString() fonksiyonunu kullanın.

bool tipli veriler "true" veya "false" dizgileriyle çıktılanır. Tarihler YYYY.MM.DD HH:MM:SS şeklinde çıktılar oluşturur. Bir tarihi farklı bir formda görüntülemek için TimeToString() fonksiyonunu kullanın. color tipli veriler ya R,G,B dizgileri şeklinde veya renk kümesinde mevcut olan bir rengin ismiyle çıktılanırlar.

Alert() function does not work in the Strategy Tester.

CheckPointer

Fonksiyon, nesne işaretçisinin tipine dönüş yapar.

```
ENUM_POINTER_TYPE  CheckPointer(
   object* anyobject      // nesne işaretçisi
   );
```

Parametreler

anyobject

[in] Nesne işaretçisi.

Dönüş değeri

ENUM_POINTER_TYPE sayımının değerlerinden birine dönüş yapar.

Not

Yanlış işaretçinin çağrılması, programda kritik sonlandırmaya yol açar. Bu yüzden bir işaretçiyi kullanmadan, CheckPointer fonksiyonunun çağrılması gerekir. Bir işaretçi şu durumlarda yanlış olabilir:

- işaretçi değeri NULL değerine eşittir;
- nesne, delete operatörü ile silinmiştir.

Bu fonksiyon işaretçinin geçerliliğini kontrol etmek için kullanılabilir. Sıfır harici bir değer, işaretçinin erişim için kullanılabilir olduğunu garanti eder.

İşaretçiyi hızlı bir şekilde doğrulamak adına, CheckPointer fonksiyonunun örtük çağrısı aracılığıyla işaretçinin geçerliliğini kontrol eden "!" operatörünü de kullanabilirsiniz (örnek).

Örnek:

```
//+------------------------------------------------------------------+
//| Listeyi, elemanlarını silerek sil                                |
//+------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- döngüde çalışması için işaretçiyi servis et
   CItem* item;
//--- döngüyü incele ve dinamik işaretçileri silmeyi dene
   while(CheckPointer(m_items)!=POINTER_INVALID)
     {
      item=m_items;
      m_items=m_items.Next();
      if(CheckPointer(item)==POINTER_DYNAMIC)
        {
         Print("Dinamik nesne ",item.Identifier()," silinecek");
         delete (item);
        }
      else Print("Dinamik olmayan nesne ",item.Identifier()," silinemez");
     }
```

```
//---
  }
```

Ayrıca Bakınız

Nesne İşaretçileri, Nesne İşaretçisinin Kontrolü, Nesne Silme Operatörü delete

Comment

Kullanıcı tarafından tanımlanmış bir yorumu çizelgenin sol üst köşesine çıktılar.

```
void  Comment(
   argument,      // ilk değer
   ...            // sonraki değerler
   );
```

Parametreler

...

[in] Virgüllerle ayrılmış herhangi bir değer. Çıktı bilgisini birkaç satıra bölmek için "\n" veya "\r\n" satır atlama işlemleri kullanılır. Parametre sayısı 64'ü geçemez. Giriş yorumlarının toplam uzunluğu (görünmeyen semboller de dahil) 2045 karakteri geçemez (fazla semboller çıktılama sırasında budanır).

Dönüş değeri

Dönüş değeri yok

Not

Diziler Comment() fonksiyonuna geçirilemez. Diziler eleman-eleman girilmelidir.

double tipi veriler, noktadan sonra 16 hanelik kesinliğe kadar, hangi notasyonun daha düzenli olduğuna bağlı olarak bilimsel veya geleneksel biçimlerde çıktılanabilir. float tipi veri, noktadan sonra 5 hane ile çıktılanır. Reel sayıları farklı bir kesinlikte veya ön tanımlı bir biçimde çıktılamak için, DoubleToString() fonksiyonunu kullanın.

bool tipi veriler "true" veya "false" dizgileriyle çıktılanırlar. Tarihler YYYY.MM.DD HH:MM:SS şeklinde gösterilir. Tarihleri başka bir biçimde göstermek için, TimeToString() fonksiyonunu kullanın. color tipi ya R,G,B dizgisi şeklinde yada renk kümesinde bulunan bir renk ismi şeklinde yazılır.

Comment() function does not work during optimization in the Strategy Tester.

Örnek:

```
void OnTick()
  {
//---
   double Ask,Bid;
   int Spread;
   Ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
   Bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   Spread=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD);
//--- Çıktı değerleri üç satırda
   Comment(StringFormat("Fiyatları göster\nAsk = %G\nBid = %G\nSpread = %d",Ask,Bid,Sp
  }
```

Ayrıca Bakınız

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
   );
```

Parametreler

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.

Dönüş değeri

Amount of bytes in the destination array or 0 in case of error. To obtain information about the error call the GetLastError() function.

Örnek:

```
//+------------------------------------------------------------------+
//| ArrayToHex                                                       |
//+------------------------------------------------------------------+
string ArrayToHex(uchar &arr[],int count=-1)
  {
   string res="";
//--- kontrol et
   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());
  }
```

Ayrıca bakınız

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
   );
```

Parametreler

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.

Dönüş değeri

Amount of bytes in the destination array or 0 in case of error. To obtain information about the error call the GetLastError() function.

Ayrıca bakınız

Array Functions, CryptEncode()

DebugBreak

Bu, hata ayıklama sırasındaki bir kırılma noktasıdır.

```
void  DebugBreak();
```

Dönüş değeri

Dönüş değeri yok.

Not

Bir MQL5 programının çalışması, sadece program hata ayıklama modunda çalıştırılmışsa kesintiye uğrayabilir. Fonksiyon değişkenlerin değerlerini görebilmek için ve/veya adım-adım çalıştırılma amacıyla kullanılabilir.

ExpertRemove

Fonksiyon bir Uzman Danışmanı durdurur veya çizelgeden kaldırır.

```
void  ExpertRemove();
```

Dönüş değeri

Dönüş değeri yok.

Not

Uzman Danışman, ExpertRemove() fonksiyonunun çağrılmasıyla hemen durmaz; Sadece Uzman Danışmanı durdurmak için bir bayrak ayarlanır. Bu daha sonraki işlemlerin uygulanmayacağı anlamına gelir, OnDeinit() çağrılacaktır ardından Uzman Danışman çizelgeden kaldırılacaktır.

Strateji sınayıcısında OnInit() yöneticisi içerisinde ExpertRemove() fonksiyonunu çağırmak, mevcut parametre kümesi üzerindeki sınama işlemini iptal eder. Böyle bir tamamlama, bir başlatma hatası olarak kabul edilir.

Bir uzman danışmanın başarılı başlatımı sonrası strateji sınayıcısında ExpertRemove() fonksiyonu çağrıldığında, bu süren test, OnDeinit() ve OnTester() fonksiyonlarının çağrılması ile normal bir şekilde sonlanır. Bu durumda, tüm alım-satım istatistikleri ve bir optimizasyon kriteri değeri elde edilir.

Örnek:

```
//+------------------------------------------------------------------+
//|                                            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;// UD yüklenmeden önceki tik sayısı
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(TimeCurrent(),": " ,__FUNCTION__," sebep kodu = ",reason);
//--- "temizle" yorumu
   Comment("");
//---
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
```

```
   static int tick_counter=0;
//---
   tick_counter++;
   Comment("\nUzman danışman kaldırılmadan önce ",__FILE__," kalmış",
           (ticks_to_close-tick_counter)," ticks");
//--- önce
   if(tick_counter>=ticks_to_close)
     {
      ExpertRemove();
      Print(TimeCurrent(),": ",__FUNCTION__," uzman danışman kaldırılacak");
     }
   Print("tick_counter =",tick_counter);
//---
  }
//+------------------------------------------------------------------+
```

Ayrıca Bakınız

Programların çalıştırılması, Müşteri terminali olayları

GetPointer

Fonksiyon nesne işaretçisine dönüş yapar.

```
void*  GetPointer(
   any_class anyobject      // herhangi bir sınıf nesnesi
   );
```

Parametreler

anyobject

[in] Herhangi bir sınıfın nesnesi.

Dönüş değeri

Fonksiyon nesne işaretçisine dönüş yapar.

Not

Sadece sınıf nesnelerinin işaretçileri vardır. Yapıların örneklerinin ve basit tipli değişkenlerin işaretçisi olamaz. new() operatörü ile oluşturulmamış bir nesne, örneğin otomatik olarak nesneler dizisinde oluşturulmuş bir nesne, yine de bir şekillendiriciye sahiptir. Ama bu işaretçi POINTER_AUTOMATIC otomatik tipinde olacaktır, bu yüzden delete() operatörü buna uygulanamaz. Bunun yanında, tip işaretçisi, POINTER_DYNAMIC tipinin dinamik işaretçilerinden farklı değildir.

Yapı tipi ve basit tipli değişkenler işaretçilere sahip olmadıklarından, GetPointer() fonksiyonunu bunlara uygulamak yasaklanmıştır. İşaretçinin fonksiyon argümanı olarak geçirilmesi de yasaklanmıştır. Bu durumların hiç birinde derleyici hata uyarısı yapmaz.

Yanış işaretçinin çağrılmasının denenmesi, programda kritik sonlanmaya yol açar. Bu yüzden, bir işaretçi kullanmadan önce CheckPointer() fonksiyonu çağrılmalıdır. Bir işaretçi şu durumlarda geçersiz olabilir:

- işaretçi değeri NULL değerine eşittir;
- nesne, delete operatörü ile silinmiştir.

Bu fonksiyon işaretçinin geçerliliğini kontrol etmek için kullanılır. Sıfır harici bir değer, işaretçinin erişim için kullanılabileceğini garanti eder.

Örnek:

```
//+------------------------------------------------------------------+
//|                                              Check_GetPointer.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                               https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

//+------------------------------------------------------------------+
//| Liste elemanını uygulayan sınıf                                  |
//+------------------------------------------------------------------+
```

```
class CItem
  {
   int               m_id;
   string            m_comment;
   CItem*            m_next;
public:
                     CItem() { m_id=0; m_comment=NULL; m_next=NULL; }
                    ~CItem() { Print("nin yıkıcısı ",m_id,
                                     (CheckPointer(GetPointer(this))==POINTER_DYNAMIC)
                                     "dinamik":"dinamik değil"); }
   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; }
  };
//+------------------------------------------------------------------+
//| Listenin basit sınıfı                                            |
//+------------------------------------------------------------------+
class CMyList
  {
   CItem*            m_items;
public:
                     CMyList() { m_items=NULL; }
                    ~CMyList() { Destroy(); }
    bool             InsertToBegin(CItem* item);
    void             Destroy();
  };
//+------------------------------------------------------------------+
//| Liste elemanını başa yerleştirme                                 |
//+------------------------------------------------------------------+
bool CMyList::InsertToBegin(CItem* item)
  {
   if(CheckPointer(item)==POINTER_INVALID) return(false);
//---
   item.Next(m_items);
   m_items=item;
//---
   return(true);
  }
//+------------------------------------------------------------------+
//| Liste elemanlarını silerek silme                                 |
//+------------------------------------------------------------------+
void CMyList::Destroy()
  {
//--- döngüde çalışması için işaretçiyi servis et
   CItem* item;
//--- döngüyü incele ve dinamik işaretçileri sil
   while(CheckPointer(m_items)!=POINTER_INVALID)
```

```
      {
       item=m_items;
       m_items=m_items.Next();
       if(CheckPointer(item)==POINTER_DYNAMIC)
         {
          Print("Dinamik nesne ",item.Identifier()," silinecek");
          delete (item);
         }
       else Print("Dinamik olmayan nesne ",item.Identifier()," silinemez");
      }
//---
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   CMyList list;
   CItem   items[10];
   CItem*  item;
//--- bir dinamik nesne işaretçisi oluştur ve listeye ekle
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(100,"dynamik");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//--- listeye otomatik işaretçiler ekle
   for(int i=0; i<10; i++)
     {
      items[i].Initialize(i,"otomatik");
      items[i].PrintMe();
      item=GetPointer(items[i]);
      if(CheckPointer(item)!=POINTER_INVALID)
         list.InsertToBegin(item);
     }
//--- listenin başına bir yada daha fazla dinamik nesne ekle
   item=new CItem;
   if(item!=NULL)
     {
      item.Initialize(200,"dynamik");
      item.PrintMe();
      list.InsertToBegin(item);
     }
//--- tüm liste elemanlarını sil
   list.Destroy();
//--- script bittikten sonra tüm liste elemanları silinecek
//--- terminaldeki Uzmanlar sekmesine bak
```

```
}
```

Ayrıca Bakınız

Nesne İşaretçileri, Nesne İşaretçisinin Kontrolü, Nesne Silme Operatörü delete

GetTickCount

GetTickCount() fonksiyonu sistem başlangıcından bu yana geçen milisaniyelerin sayısını verir.

```
uint  GetTickCount();
```

Dönüş değeri

uint tipi değer.

Not

Sayaç, sistem saatinin kısıtlamalarına bağlı olarak sınırlanmıştır. Zaman, bir işaretsiz tamsayı şeklinde saklanır, bu nedenle bilgisayarın kesintisiz çalışması durumunda her 49.7 günde bir yeniden başlar.

Örnek:

```
#define MAX_SIZE 40
//+------------------------------------------------------------------+
//| 40 Fibonacci sayısının hesaplama süresini ölçen script           |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- İlk değeri hatırla
   uint start=GetTickCount();
//--- Fibonacci serisindeki sonraki sayıyı almak için bir değişken
   long fib=0;
//--- Döngü içinde, Fibonacci serinden belirlenen miktarda sayıyı hesapla
   for(int i=0;i<MAX_SIZE;i++) fib=TestFibo(i);
//--- Harcanan zamanı milisaniye olarak göster.
   uint time=GetTickCount()-start;
//--- Uzmanlar bültenine bir mesaj çıktıla
   PrintFormat("İlk %d Fibonacci sayısının hesaplanması %d ms zaman aldı",MAX_SIZE,tir
//--- Script tamamlandı
   return;
  }
//+------------------------------------------------------------------+
//| Fibonacci sayısını serideki indisiyle almak için bir fonksiyon   |
//+------------------------------------------------------------------+
long TestFibo(long n)
  {
//--- Fibonacci serisinin ilk elemanı
   if(n<2) return(1);
//--- Tüm diğer elemanlar şu formülle hesaplanır
   return(TestFibo(n-2)+TestFibo(n-1));
  }
```

Ayrıca Bakınız

Tarih ve Zaman, GetTickCount64, GetMicrosecondCount

GetTickCount64

GetTickCount64() fonksiyonu, sistemin başlatılmasından bu yana geçen milisaniye sayısını geri döndürür.

```
ulong  GetTickCount64();
```

Geri dönüş değeri

ulong tipi değer.

Not

Sayaç, genellikle 10-16 milisaniye hassasiyetle sonuç veren sistem saatinin doğruluğuyla sınırlıdır. uint tipinde olan ve sürekli bilgisayar çalışması durumunda içeriği her 49,7 günde bir yeniden başlayan GetTickCount'tan farklı olarak, GetTickCount64() sınırsız bilgisayar çalışma süresi için kullanılabilir ve yeniden başlamaya maruz kalmaz.

Ayrıca bakınız

Tarih ve Zaman, EventSetMillisecondTimer, GetTickCount, GetMicrosecondCount

GetMicrosecondCount

The GetMicrosecondCount() function returns the number of microseconds that have elapsed since the start of MQL5-program.

```
ulong  GetMicrosecondCount();
```

Dönüş değeri

ulong tipi değer.

Örnek:

```
//+------------------------------------------------------------------+
//| Test function                                                    |
//+------------------------------------------------------------------+
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++;
     }
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   uint   ui=0,ui_max=0,ui_min=INT_MAX;
   ulong  ul=0,ul_max=0,ul_min=INT_MAX;
//--- number of measurements
   for(int count=0;count<1000;count++)
     {
      uint  ui_res=0;
      ulong ul_res=0;
      //---
      for(int n=0;n<2;n++)
        {
         //--- select measurement type
         if(n==0) ui=GetTickCount();
         else     ul=GetMicrosecondCount();
         //--- execute code
         Test();
         //--- add measurement result (depending on type)
```

```
         if(n==0) ui_res+=GetTickCount()-ui;
         else     ul_res+=GetMicrosecondCount()-ul;
        }
      //--- calculate minimum and maximum time for both measurements
      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))
  }
```

Ayrıca Bakınız

Tarih ve Zaman, EventSetMillisecondTimer, GetTickCount, GetTickCount64

MessageBox

Bir mesaj kutusunu oluşturur, gösterir ve yönetir. Bir mesaj kutusu, bir mesaj, bir başlık ve ön tanımlı işaretlerin ve komut satırlarının bir kombinasyonundan oluşur.

```
int  MessageBox(
   string  text,              // mesaj metni
   string  caption=NULL,      // kutu başlığı
   int     flags=0            // kutudaki düğmeler kümesini belirler
   );
```

Parametreler

text

[in] Çıktılanacak mesaj metni.

caption=NULL

[in] Kutu başlığında görüntülenecek isteğe bağlı metin. Eğer bu parametre boşsa, kutu başlığında Uzman Danışmanın ismi gösterilir.

flags=0

[in] Mesaj kutusunun görünümünü ve davranışını belirleyen isteğe bağlı bayraklar. Bayraklar, özel bir bayrak grubunun kombinasyonu olabilir.

Dönüş değeri

fonksiyon başarıyla gerçekleşmişse dönüş değeri, MessageBox() dönüş kodlarından biri olacaktır.

Not

MessageBox() çağrısı, kullanıcının yanıtını beklerken tüm zaman boyunca yürütme iş parçacığını askıya aldığından, fonksiyon özel göstergelerde kullanılamaz. Her bir sembol için tüm göstergeler tek bir iş parçacığında yürütüldüğünden, bu tür bir askıya alma, bu sembol için tüm zaman aralıklarında tüm grafiklerin çalışmasını imkansız hale getirir.

MessageBox() function does not work in the Strategy Tester.

PeriodSeconds

Periyot içindeki saniye sayısına dönüş yapar.

```
int  PeriodSeconds(
   ENUM_TIMEFRAMES  period=PERIOD_CURRENT      // çizelge periyodu
   );
```

Parametreler

period=PERIOD_CURRENT

[in] ENUM_TIMEFRAMES sayımından çizelge periyodunun değeri. Eğer parametre belirtilmemişse, programın çalıştığı mevcut çizelge periyodunun saniye sayısına dönüş yapar.

Dönüş değeri

Seçilmiş periyot içindeki saniye değeri.

Ayrıca Bakınız

_Period, Çizelge Zaman Aralıkları, Tarih ve Zaman, Nesnelerin Görünürlüğü

PlaySound

Bir ses dosyasını oynatır.

```
bool  PlaySound(
   string  filename      // dosya ismi
   );
```

Parametreler

filename

[in] Ses dosyasının konumu. If filename=NULL, the playback is stopped.

Dönüş değeri

Dosya bulunmuşsa 'true', aksi durumda 'false'.

Not

Dosya, terminal_dizini\Sounds konumuna veya bunun bir alt konumuna yerleştirilmiş olmalıdır. Sadece WAV dosyaları oynatılır.

Call of PlaySound() with NULL parameter stops playback.

PlaySound() function does not work in the Strategy Tester.

Ayrıca Bakınız

Kaynaklar

Print

Uzman Danışman günlüğüne bir mesaj girer. Parametreler herhangi bir tipte olabilirler.

```
void  Print(
   argument,      // ilk değer
   ...            // sonraki değerler
   );
```

Parametreler

...

[in] Virgül ile ayrılmış herhangi bir değer. Parametre sayısı 64'ü geçemez.

Not

Diziler Print() fonksiyonuna geçirilemez. Diziler elemen eleman girilmelidir.

double tipi veriler, noktadan sonra 16 hanelik kesinliğe kadar gösterilebilir ve hangi girdinin daha düzenli olduğuna bağlı olarak bilimsel veya geleneksel biçimlerde çıktılanabilir. float tipi veri, noktadan sonra 5 hane ile çıktılanır. Reel sayıları önceden tanımlanmış bir biçimde veya farklı bir kesinlikte çıktılamak için, PrintFormat() fonksiyonunu kullanın.

bool tipi veriler, "true" veya "false" dizgileri ile çıktılanırlar. Tarihler YYYY.MM.DD HH:MM:SS şeklinde gösterilir. Veriyi farklı bir biçimde göstermek için TimeToString()fonksiyonunu kullanın. color tipli verilere ya R,G,B dizgileri şeklinde veya renk kümesinde mevcut olan bir rengin ismiyle dönüş yapılır.

Print() function does not work during optimization in the Strategy Tester.

Örnek:

```
void OnStart()
  {
//--- DBL_MAX değerini Print() kullanarak çıktıla, bu şuna eşdeğerdir PrintFormat(%%.1
   Print("---- DBL_MAX nasıl görünüyor -----");
   Print("Print(DBL_MAX)=",DBL_MAX);
//--- Şimdi PrintFormat() kullanarak bir DBL_MAX sayısını çıktıla
   PrintFormat("PrintFormat(%%.16G,DBL_MAX)=%.16G",DBL_MAX);
//--- Uzmanlar bültenine çıktıla
// Print(DBL_MAX)=1.797693134862316e+308
// PrintFormat(%.16G,DBL_MAX)=1.797693134862316E+308

//--- float tipinin nasıl çıktılandığına bak
   float c=(float)M_PI; // Hedef tipe açık olarak dönüştürmeliyiz
   Print("c=",c, "    Pi=",M_PI, "    (float)M_PI=",(float)M_PI);
// c=3.14159    Pi=3.141592653589793    (float)M_PI=3.14159

//--- Reel tiplerin aritmetik işlemlerinde ne olabilir göster
   double a=7,b=200;
   Print("---- Aritmetik işlemlerden önce");
   Print("a=",a,"   b=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
```

```
//--- a'yı b'ye böl (7/200)
   a=a/b;
//--- Şimdi b değişkeni içine yalandan bir değer kaydet
   b=7.0/a; // Beklenen şu b=7.0/(7.0/200.0)=>7.0/7.0*200.0=200 - ama farklı
//--- b'nin yeni hesaplanan değerini çıktıla
   Print("----- Aritmetik işlemlerden sonra");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- Uzmanlar bültenine çıktıla
// Print(b)=200.0
// Print(DoubleToString(b,16))=199.9999999999999716 (b'nin atrk 200.0 olmadığını gör)

//--- çok küçük bir değer oluştur, epsilon=1E-013
   double epsilon=1e-13;
   Print("---- Çok küçük bir değer oluştur");
   Print("epsilon=",epsilon); // epsilon=1E-013 değerini al
//--- Şimdi epsilonu b'den çıkar ve Uzmanlar günlüğünde yeniden çıktıla
   b=b-epsilon;
//--- İki yol kullan
   Print("---- epsilonu b değişkeninden çıkardıktan sonra");
   Print("Print(b)=",b);
   Print("Print(DoubleToString(b,16))=",DoubleToString(b,16));
//--- Uzmanlar bültenine çıktıla
// Print(b)=199.9999999999999  (artık epsilonu çıkarmadan sonra b'nin değeri 200'e yuv
// Print(DoubleToString(b,16))=199.9999999999998578
//    (artık epsilonu çıkarmadan sonra b'nin değeri 200'e yuvarlanamaz)
  }
```

Ayrıca Bakınız

DoubleToString, StringFormat

PrintFormat

Uzman Danışman günlüğündeki sembol kümelerini ve değerlerini, ön ayarlı şekle göre biçimlendirerek girişini yapar.

```
void  PrintFormat(
   string format_string,    // biçim dizgisi
   ...                      // basit tipli değerler
   );
```

Parametreler

format_string

[in] Bir biçim dizgisi basit sembolleri içerir, eğer argümanlarla takip ediliyorsa, o zaman biçim özeliklerini de içerir.

...

[in] Virgülle ayrılmış tüm basit tipli değerler. Toplam parametre sayısı, biçim dizgisi de dahil olmak üzere 64'ü geçemez.

Dönüş değeri

Dizgi.

Not

PrintFormat() function does not work during optimization in the Strategy Tester.

Parametrelerin sayısı, sırası ve tipi, niteleyicilerin kümesiyle bir bir örtüşmelidir, aksi durumda çıktılama sonucu tanımsızdır. PrintFormat() yerine printf() kullanabilirsiniz.

Eğer biçim dizgisi parametrelerce takip ediliyorsa, bu parametrelerin çıktı biçimlerini belirleyen biçim özelliklerini içermelidir. Biçimin belirlemesi daima yüzde (%) işareti ile başlar.

Bir biçim dizgisi soldan sağa doğru okunur. İlk biçim özelliği karşılandığında (eğer varsa), biçim dizgisinden sonra gelen ilk parametrenin değeri ön ayarlı özelliğe göre dönüştürülür ve çıktılanır. İkinci biçim özelliği, ikinci parametreyi dönüşüm ve çıktılama için çağırır ve bu durum biçim dizgisinin sonuna kadar devam eder.

Biçim özelliği şu formdadır:

%[bayraklar][genişlik][.çözünürlük][{h | l | ll | I32 | I64}]tip

Biçim özelliğinin her bir alanı ya basit bir semboldür, ya da basit biçim seçeneğini ifade eden bir sayıdır. En basit biçim özelliği sadece yüzde (%) işaretini ve çıktı parametresinin tipini tanımlayan bir sembolünü (örneğin, %s) içerir. Eğer yüzde işaretini biçim dizgisinin içinde çıktılamanız gerekiyorsa %% biçimini kullanın.

bayraklar

Bayrak	Açıklama	Varsayılan Davranış
- (eksi)	Ayarlı genişlik içindeki sol hizalama	Sağ hizalama
+ (artı)	İşaret tipleri için + veya - işaretinin çıktısı	İşaret sadece değer negatif ise gösterilir
0 (sıfır)	Sıfırlar, ön ayarlı genişlik içindeki bir çıktı değerinden önce eklenir. If **0** bayrak tamsayı biçiminde belirlenir (**i**, **u**, **x**, **X**, **o**, **d**) ve kesinlik (çözünürlük) özelliği ayarlanır (örneğin, %04.d), sonrasında sıfır gözardı edilir.	Hiçbir şey eklenmez
boşluk	Boşluk, çıktı değerinden önce gösterilir, eğer çıktı işaretli ve pozitif bir değerse	Boşluklar girilmez
#	Eğer; **o**, **x** veya **X** biçimleriyle beraber kullanılıyorsa, çıktı değerinden önce sırasıyla 0, 0x veya 0X eklenir.	Hiçbir şey eklenmez
	Eğer; **e**, **E**, **a** veya **A** biçimleriyle beraber kullanılıyorsa, değer her zaman bir ondalık nokta ile gösterilir.	Ondalık nokta, sadece sıfır harici bir kısım varsa gösterilir.
	Eğer **g** veya **G** biçimleriyle beraber kullanılıyorsa; bayrak, çıktı değerinde bir ondalık noktanın varlığını tanımlar ve sıfırların atılmasını önler. # bayrağı: **c**, **d**, **i**, **u**, **s** biçimleri ile beraber kullanılırken gözardı edilir.	Ondalık nokta, sadece sıfır harici bir kısım varsa gösterilir. Ön sıfırlar atılır.

genişlik

Biçimlendirilmiş değerin çıktı sembollerinin minimal sayısını ayarlayan, negatif olmayan bir reel sayı. Eğer çıktı sembollerinin sayısı belirlenen genişlikten az ise, karşılık gelen sayıda boşluk, hizalamaya bağlı olarak (bayrak -) sağdan veya soldan eklenir. Eğer sıfır bayrağı (0) varsa, karşılık gelen sayıda sıfır, çıktı değerinden önce eklenir. Eğer çıktı sembollerinin sayısı belirlenen genişlikten fazla ise, çıktı değeri asla budanmaz.

Eğer bir yıldız (*) genişlik olarak belirlenmişse, geçirilen parametreler listesinde karşılık gelen yerde, tamsayı tipli değer belirtilmelidir. Bu, çıktı değerinin genişliğini belirlemek için kullanılacaktır.

çözünürlük

Çıktı kesinliğini ayarlayan ve negatif olmayan ondalık basamak sayısı. Genişlik özelliğinden farklı olarak çözünürlük özelliği, kesirli tipin bir kısmını yuvarlamadan atabilir.

Çözünürlük özelliğinin kullanımı, farklı biçim tipleri için farklılık gösterir.

Tipler	Açıklama	Varsayılan Davranış
a, A	Kesinlik özelliği, noktadan sonraki basamak sayısını ayarlar.	Varsayılan kesinlik – 6.

Tipler	Açıklama	Varsayılan Davranış
c, C	Kullanılmaz	
d, i, u, o, x, X	Minimal çıktı basamağı sayısını belirler. Eğer karşılık gelen bir parametredeki basamak sayısı bu kesinlikten az ise, çıktı değerinin sonuna sıfırlar eklenir. Eğer çıktı basamakları, belirtilen kesinlikten büyükse o zaman çıktı değerinde bir budama yapılmaz.	Varsayılan kesinlik - 1.
e, E, f	Noktadan sonraki çıktı basamaklarının sayısını ayarlar. Son basamak yuvarlanır.	Varsayılan kesinlik - 6. Eğer ayarlı kesinlik 0 ise ondalık kısım yoksa, ondalık nokta gösterilmez.
g, G	Anlamlı rakamların maksimal sayısını ayarlar.	6 anlamlı rakam çıktılanır.
s	Bir dizginin çıktı sembollerinin sayısını ayarlar. Eğer dizgi uzunluğu ayarlanmış kesinlik değerini geçiyorsa, dizgi budanır.	Tüm dizgi çıktılanır.

```
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

Parametre olarak geçirilen veri büyüklüklerinin belirlenmesi.

Parametre Tipi	Kullanılan Önek	Tipin birleşik belirteci
int	l (küçük L)	d, i, o, x veya X
uint	l (küçük L)	o, u, x veya X
long	ll (iki küçük L)	d, i, o, x veya X
short	h	d, i, o, x veya X
ushort	h	o, u, x veya X
int	I32	d, i, o, x veya X

Parametre Tipi	Kullanılan Önek	Tipin birleşik belirteci
uint	I32	o, u, x veya X
long	I64	d, i, o, x veya X
ulong	I64	o, u, x veya X

type

Tip belirteci, biçimlendirilmiş çıktı için tek zorunluluktur.

Sembol	Tip	Çıktı Biçimi
c	int	short tipli sembol (Unicode)
C	int	char tipli sembol (ANSI)
d	int	İşaretli ondalık tamsayı
i	int	İşaretli ondalık tamsayı
o	int	İşaretsiz sekizli tamsayı
u	int	İşaretsiz ondalık tamsayı
x	int	İşaretsiz onaltılık tamsayı, "abcdef" kullanarak
X	int	İşaretsiz onaltılık tamsayı, "ABCDEF" kullanarak
e	double	[-] d.dddde[sign] ddd biçiminde bir reel değer, burada d - tek ondalık haneyi, dddd - bir veya daha fazla ondalık haneyi, ddd - üs büyüklüğünü belirleyen üç basamaklı bir sayıyı, sign - artı veya eksiyi belirtir
E	double	e biçimiyle benzerdir ama üssün işareti büyük E ile çıktılanır (e yerine **E**)
f	double	[-] dddd.dddd biçimli bir reel sayı, burada dddd - bir veya daha fazla ondalık hanedir. Noktadan önce gösterilen basamakların sayısı, sayı değerine bağlıdır. Noktadan sonra gösterilen basamakların sayısı, istenen kesinliğe bağlıdır.
g	double	Hangisinin daha düzenli olduğuna bağlı olarak **f** veya **e** biçiminde çıktılanan bir reel değer.
G	double	Hangisinin daha düzenli olduğuna bağlı olarak **F** veya **E** biçiminde çıktılanan bir reel değer.
a	double	[-]0xh.hhhh **p**±dd biçiminde bir reel sayı, burada h.hhhh - "abcdef" kullanarak onaltılık formda mantisi, dd - üssün bir veya daha fazla hanesini belirtmektedir. Ondalık hanelerin sayısı kesinlik özelliği ile belirlenir

Sembol	Tip	Çıktı Biçimi
A	double	[-]0xh.hhhh **P**±dd biçiminde bir reel sayı, brada h.hhhh - "ABCDEF" kullanarak onaltılık formda mantisi, dd - üssün bir veya daha fazla hanesini belirtmektedir. Ondalık hanelerin sayısı kesinlik özelliği ile belirlenir
s	string	Dizgi çıktısı

PrintFormat() yerine printf() kullanabilirsiniz.

Örnek:

```
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
*/
  }
```

Ayrıca Bakınız

StringFormat, DoubleToString, Reel tipler (double, float)

ResetLastError

Önceden tanımlanmış _LastError değişkeninin değerini sıfıra ayarlar.

```
void  ResetLastError();
```

Dönüş değeri

Dönüş değeri yok.

Not

GetLastError() fonksiyonunun _LastError değişkenini sıfırlamadığı not edilmelidir. Genellikle ResetLastError() fonksiyonu hata görünümü kontrol edildikten sonra, başka bir fonksiyon çağrılmadan önce çağrılır.

ResourceCreate

Bir veri setini temel alarak, bir görüntü kaynağı oluşturur. Fonksiyonun iki çeşidi vardır:

Bir dosya temelinde, bir kaynak yaratan

```
bool  ResourceCreate(
   const string      resource_name,       // Kaynak ismi
   const string      path                 // Dosya için göreli bir konum
   );
```

Bir pikseller dizisi temelinde, bir kaynak oluşturan

```
bool  ResourceCreate(
   const string      resource_name,       // Kaynak ismi
   const uint&       data[],              // Dizi şeklinde veri seti
   uint              img_width,           // Görüntü kaynağının genişliği
   uint              img_height,          // Görüntü kaynağının yüksekliği
   uint              data_xoffset,        // Görüntünün sol üst köşesinin sağa doğru y
   uint              data_yoffset,        // Görüntünün sol üst köşesinin aşağı doğru
   uint              data_width,          // Veri seti temelinde toplam görüntü genişl
   ENUM_COLOR_FORMAT color_format         // Renk işleme yöntemi
   );
```

Parametreler

resource_name

[in] Kaynak ismi.

data[][]

[in] Bütün bir görüntü oluşturmak için tek veya iki boyutlu dizi

img_width

[in] Kaynağa yerleştirilecek dik dörtgen şekilli görüntü alanının pikseller bazında genişliği. Bu, *data_width* değerinden *büyük olamaz.*

img_height

[in] Kaynağa yerleştirilecek dik dörtgen şekilli görüntü alanının pikseller bazında yüksekliği.

data_xoffset

[in] Görüntünün dikdörtgensel alanının sağa doğru yatay konumu.

data_yoffset

[in] Görüntünün dikdörtgensel alanının aşağı doğru dikey konumu.

data_width

[in] Sadece tek boyutlu dizilerde istenir. Veri setinden görüntünün tam genişliğini ifade eder. *data_width*=0 ise, *img_width'e eşit olduğu kabul edilir.* İki boyutlu dizilerde bu parametre gözardı edilir ve *data[]* dizisinin ikinci boyutuna eşit olduğu kabul edilir.

color_format

[in] Renk işleme yöntemi, ENUM_COLOR_FORMAT sayımının değerlerinden biri olabilir.

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata ile ilgili bilgi almak için GetLastError() fonksiyonunu çağırın. Şu hatalar gerçekleşebilir:

- 4015 – ERR_RESOURCE_NAME_DUPLICATED (dinamik ve statik kaynak isimlerinin aynı olması)
- 4016 – ERR_RESOURCE_NOT_FOUND (kaynak bulunamadı)
- 4017 – ERR_RESOURCE_UNSUPPORTED_TYPE (bu kaynak tipi desteklenmiyor)
- 4018 – ERR_RESOURCE_NAME_IS_TOO_LONG (kaynak ismi çok uzun)

Not

Fonksiyonun ikinci versiyonu, aynı kaynağı farklı yükseklik, genişlik ve kaydırma parametreleriyle oluşturmak için çağrılırsa, yeni bir kaynak oluşturmayacak, aksine var olanı güncelleyecekir.

Fonksiyonun ilk versiyonu dosyalardan görüntü ve ses yüklemek için kullanılır, ikinci versiyonu ise sadece görüntülerin dinamik olarak oluşturulması için kullanılır.

Görüntüler, 24 veya 32 bitlik renk derinliği ile BMP biçiminde olmalıdır. Sesler sadece WAV biçiminde olabilir. Kaynak büyüklüğü 16 Mb'i aşmamalıdır.

ENUM_COLOR_FORMAT

Tanımlayıcı	Açıklama
COLOR_FORMAT_XRGB_NOALPHA	Alfa kanalı bileşeni gözardı edilir
COLOR_FORMAT_ARGB_RAW	Renk bileşenleri terminal tarafından işlenmez (kullanıcı tarafından düzgün şekilde ayarlanmalıdır)
COLOR_FORMAT_ARGB_NORMALIZE	Renk bileşenleri terminal tarafından işlenir

Ayrıca Bakınız

Kaynaklar, ObjectCreate(), ObjectSetString(), OBJPROP_BMPFILE

ResourceFree

Fonksiyon dinamik olarak oluşturulmuş kaynağı (bunun için ayrılmış hafızayı boşaltarak) siler.

```
bool  ResourceFree(
   const string  resource_name      // kaynak ismi
   );
```

Parametreler

resource_name

[in] Kaynak ismi "::" ile başlamalıdır.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

Not

ResourceFree() fonksiyonu kaynaklarla aktif olarak çalışılırken, MQL5 uygulama geliştiricilerinin bellek tüketimini kontrol etmelerini sağlar. Bellekten silinen kaynağa bağlı grafiksel nesneler silinme bittikten sonra düzgün şekilde gözükecektir. Ama yeni oluşturulan grafiksel nesneler (OBJ_BITMAP ve OBJ_BITMAP_LABEL) silinmiş kaynağı kullanamayacaklardır.

Fonksiyon, sadece program tarafından oluşturulmuş dinamik kaynakları siler.

Ayrıca Bakınız

Kaynaklar, ObjectCreate(), PlaySound(), ObjectSetString(), OBJPROP_BMPFILE

ResourceReadImage

Fonksiyon, ResourceCreate() fonksiyonu ile oluşturulmuş veya derleme sırasında EX5 dosyasına kaydedilmiş grafiksel kaynaklardan veri alır.

```
bool  ResourceReadImage(
   const string      resource_name,       // okunacak grafiksel kaynak ismi
   uint&             data[],              // kaynaktan veri alacak dizi
   uint&             width,               // kaynaktaki görüntü genişliğini almak için
   uint&             height,              // kaynaktaki görüntü yüksekliğini almak iç
   );
```

Parametreler

resource_name

[in] Bir görüntü içeren grafiksel kaynağın ismi. Kendi kaynaklarına erşim sağlamak için, isim kısa biçimde kullanılır "::resourcename". Eğer derlenmiş bir EX5 dosyasından kaynak indiriyorsak, dosyanın tam ismi, MQL5 konumuna, dosyaya ve kaynak isimlerine göre belirlenmiş bir adresle kullanılmalıdır - "dosya_yolu\\dosyaismi.ex5::kaynakismi".

data[][]

[in] Grafiksel kaynaktan veri almak için bir veya iki boyutlu dizi.

img_width

[out] Grafiksel kaynak görüntüsünün piksel olarak genişliği.

img_height

[out] Grafiksel kaynak görüntüsünün piksel olarak yüksekliği.

Dönüş değeri

Başarılı ise 'true', değilse 'false'. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

Not

Eğer *data[]* dizisi daha sonra bir grafiksel kaynak oluşturmak için kullanılacaksa, COLOR_FORMAT_ARGB_NORMALIZE veya COLOR_FORMAT_XRGB_NOALPHA renk biçimleri kullanılmalıdır.

Eğer *data[]* dizisi iki boyutluysa ve ikinci boyutu X (genişliği) grafiksel kaynağın büyüklüğünden küçükse, ResourceReadImage() fonksiyonu false değerine dönüş yapar ve okuma gerçekleşmez. Ama eğer kaynak mevcutsa gerçek görüntü büyüklüğü, genişlik ve yükseklik parametrelerine döndürülür. Bu, kaynaktan veri almak için başka bir denemeye izin verecektir.

Ayrıca Bakınız

Kaynaklar, ObjectCreate(), ObjectSetString(), OBJPROP_BMPFILE

ResourceSave

Bir kaynağı belirtilen bir dosyaya kaydeder.

```
bool  ResourceSave(
   const string  resource_name      // Kaynak ismi
   const string  file_name          // Dosya ismi
   );
```

Parametreler

resource_name

[in] Kaynağın ismi "::" ile başlamalıdır.

file_name

[in] Dosyanın, MQL5\Files yoluna göre ismi.

Dönüş değeri

Başarı durumunda 'true', aksi durumda 'false'. Hata bilgisi için GetLastError() fonksiyonunu çağırın.

Not

Fonksiyon her zaman dosyanın üzerine yazar ve eğer gerekliyse tüm istenen orta seviye dosya yollarını, dosya isminin içinde oluşturur.

Ayrıca Bakınız

Kaynaklar, ObjectCreate(), PlaySound(), ObjectSetString(), OBJPROP_BMPFILE

SetReturnError

Operasyon tamamlanırken terminal işlemini geri döndüren kodu ayarlar.

```
void  SetReturnError(
   int ret_code      // müşteri terminali tamamlanma kodu
   );
```

Parametreler

ret_code

[in] Operasyon tamamlanırken müşteri terminali işlemi tarafından geri döndürülecek kod.

Geri dönüş değeri

Geri dönüş değeri yok.

Not

SetReturnError() fonksiyonunu kullanarak *ret_code* geri dönüş kodunu ayarlamak komut satırı aracılığıyla terminal başlatıldığında programa ait operasyon tamamlanma nedenlerini analiz etmede kullanışlıdır.

TerminalClose() un aksine; SetReturnError(), terminal operasyonu tamamlamaz. Bunun yerine, yalnızca operasyon tamamlandıktan sonra terminal işlemini döndüren kodu ayarlar.

Eğer SetReturnError() işlevi birden çok kez ve/veya farklı MQL5 programlarından çağrılırsa, terminal son ayarlanmış dönüş kodunu geri döndürür.

Belirlenmiş kod, aşağıdaki durumlar dışında terminal işlemi tamamlandığında geri döndürülür:

- gerçekleşim sırasında bir kritik hata meydana geldiğinde;
- Belirli bir kod ile terminal işlemi tamamlama komutunu veren TerminalClose(int ret_code) fonksiyonu çağrıldığında.

Ayrıca bakınız

Program Yürütme, Program Hatası, Sonlandırma Neden Kodları, TerminalClose

SetUserError

Ön tanımlı _LastError değişkenini ERR_USER_ERROR_FIRST + user_error toplamına eşit olan değere ayarlar

```
void  SetUserError(
   ushort user_error,    // hata numarası
   );
```

Parametreler

user_error

[in] Hata bir kullanıcı tarafından ayarlanmış numara.

Dönüş değeri

Dönüş değeri yok.

Not

SetUserError(user_error) fonksiyonu ile bir hata ayarlandıktan sonra, GetLastError() fonksiyonu ERR_USER_ERROR_FIRST + user_error toplamına eşit olan değere dönüş yapar

Örnek:

```
void OnStart()
  {
//--- hata numarası ayarla 65537=(ERR_USER_ERROR_FIRST +1)
   SetUserError(1);
//--- son hata kodunu al
   Print("GetLastError = ",GetLastError());
/*
   Result
   GetLastError = 65537
*/
  }
```

Sleep

Fonksiyon, mevcut Uzman Danışmanın veya betiğin çalışmasını belirli bir aralık için durdurur.

```
void  Sleep(
   int  milliseconds      // aralık
   );
```

Parametreler

milliseconds

[in] Milisaniye bazında gecikme aralığı.

Dönüş değeri

Dönüş değeri yok.

Not

Sleep() fonksiyonu özel göstergelerde çağrılamaz; göstergeler arayüz iş parçacığının içinde çalıştırılır ve bunu yavaşlatmamalıdırlar. Fonksiyon her 0.1 saniyede için, Uzman Danışman durma bayrağının gömülü kontrolüne sahip olur.

TerminalClose

Fonksiyon, terminale işlemi tamamlama komutu verir.

```
bool  TerminalClose(
   int ret_code      // müşteri terminalinin kapanma kodu
   );
```

Parametreler

ret_code

[in] Dönüş kodu, işlemin tamamlanmasıyla, müşteri terminalinin işlemiyle döndürülür.

Dönüş değeri

Başarılı sonuç için 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Not

TerminalClose() fonksiyonu terminali hemen durdurmaz, sadece terminale işlemi tamamlaması için komut verir.

TerminalClose() çağrısını yapan Uzman Danışman kodu, ani tamamlama için tüm gerekli ayarlamalara sahip olmalıdır (örneğin, tüm açık dosyalar normal şekilde kapatılmalıdır). Bu fonksiyonun çağrısının ardından return operatörü gelmelidir.

ret_code parametresi, terminal işleminin program sonlandırılma sebeplerini analiz etmek için, gerekli dönüş kodunun - sonlandırma komut iletisinden başlatıldığında - belirtilmesini sağlar.

Örnek:

```
//--- giriş parametreleri
input int  tiks_before=500; // sonlandırma anına kadar tik sayısı
input int  pips_to_go=15;   // pip bazında uzaklık
input int  seconds_st=50;   // Uzman Danışmana verilen saniyelerin sayısı
//--- globaller
datetime   launch_time;
int        tick_counter=0;
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Print(__FUNCTION__," sebep kodu = ",reason);
   Comment("");
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
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;
     }
//--- pip bazında fiyat mesafesi
   distance=(tick.bid-first_bid)/_Point;
//--- UD işlemini takip etmek için bir bildirim göster
   string comm="Başlangıç anından:\r\n\x25CF geçen saniye: "+
               IntegerToString(tick.time-launch_time)+" ;"+
               "\r\n\x25CF tikler alındı: "+(string)tick_counter+" ;"+
               "\r\n\x25CF fiyatı şu kadar nokta gitti: "+StringFormat("%G",distance);
   Comment(comm);
//--- terminalin kapanma koşulunun kontrol edildiği bölüm
   if(tick_counter>=tiks_before)
      TerminalClose(0);    // tik sayacıyla çıkış
   if(distance>pips_to_go)
      TerminalClose(1);    // pips_to_go pip sayısı kadar yukarı çık
   if(distance<-pips_to_go)
      TerminalClose(-1);   // pips_to_go pip sayısı kadar aşağı in
   if(tick.time-launch_time>seconds_st)
      TerminalClose(100);  // zaman aşımı ile sonlandırma
//---
  }
```

Ayrıca Bakınız

Program çalıştırma, Çalıştırma hataları, Sonlandırma sebepleri

TesterHideIndicators

Bir EA'da kullanılan displaying/hiding göstergelerinin modunu ayarlar. Fonksiyon, yalnızca test sırasında kullanılan göstergelerin görünürlüğünü yönetmek için tasarlanmıştır.

```
void  TesterHideIndicators(
   bool      hide     // bayrak
   );
```

Parameters

hide

[in] Test ederken göstergeleri gizlemek için bayrak. Oluşturulan göstergeleri gizlemek için true, aksi halde false ayarlar.

Dönen değer

Yok.

Not

Varsayılan olarak, test edilmiş bir EA'da oluşturulan tüm göstergeler görsel test şemasında görüntülenir. Ayrıca, bu göstergeler, test tamamlandığında otomatik olarak açılan grafikte gösterilir. TesterHideIndicators() fonksiyonu, geliştiricilerin kullanılan göstergelerin gösterimini devre dışı bırakma yeteneğini uygulamasına olanak tanır.

Bir EA test ederken uygulanan bir göstergenin gösterimini devre dışı bırakmak için, EA'nın handlesini oluşturmadan önce TesterHideIndicators() fonksiyonunu çağırın - bundan sonra oluşturulan tüm göstergeler gizli bir bayrak ile işaretlenir. Bu göstergeler, görsel bir test sırasında ve testin tamamlanmasından sonra otomatik olarak açılan grafikte gösterilmez.

Yeni oluşturulan göstergelerin gizleme modunu devre dışı bırakmak için TesterHideIndicators() öğesini false değerine eşit olarak çağırın. Test edilmiş EA'dan doğrudan üretilen göstergeler test çizelgesinde görüntülenebilir. Bu kural <data_folder>MQL5\Profiles\Templates içinde tek bir şablon bulunmadığında geçerlidir.

Eğer <data_folder>MQL5\Profiles\Templates rehberi özel bir şablon <EA_name>.tpl içeriyorsa, sadece bu şablondan gelen göstergeler, görsel bir test sırasında ve test çizelgesinde görüntülenir. Bu durumda, test edilen EA'da uygulanan göstergeler gösterilmez. Bu davranış, TesterHideIndicators() true'ya eşit olsa bile EA kodunda çağırılır.

Eğer <data_folder>MQL5\Profiles\Templates rehberi tester.tpl yerine özel bir <EA_name>.tpl şablonu içermiyorsa, tester.tpl'deki indikatörler ve EA'dakiler gösterilmez, TesterHideIndicators() işlevi tarafından görsel bir test sırasında ve test çizelgesinde gösterilir. Eğer Tester.tpl şablonu yoksa, bunun yerine default.tpl şablonundaki göstergeler kullanılır.

Eğer strateji tester uygun bir şablon (<EA_name>.tpl, tester.tpl or default.tpl) bulmazsa, EA'da uygulanan göstergelerin görüntülenmesi TesterHideIndicators() fonksiyonu tarafından tamamen yönetilmektedir.

Örnek:

```
bool CSampleExpert::InitIndicators(void)
  {
   TesterHideIndicators(true);
```

```
//--- MACD indikatörü oluşturma
   if(m_handle_macd==INVALID_HANDLE)
      if((m_handle_macd=iMACD(NULL,0,12,26,9,PRICE_CLOSE))==INVALID_HANDLE)
        {
         printf("Error creating MACD indicator");
         return(false);
        }
   TesterHideIndicators(false);
//--- EMA indikatörü oluşturma ve onu toplama
   if(m_handle_ema==INVALID_HANDLE)
      if((m_handle_ema=iMA(NULL,0,InpMATrendPeriod,0,MODE_EMA,PRICE_CLOSE))==INVALID_H
        {
         printf("Error creating EMA indicator");
         return(false);
        }
//--- başarılı
   return(true);
  }
```

Ayrıca bakın

IndicatorRelease

TesterStatistics

Sınama sonuçları temelinde hesaplanan istatistiksel parametre değerine dönüş yapar.

```
double  TesterStatistics(
   ENUM_STATISTICS statistic_id      // Tanımlayıcı
   );
```

Parametreler

statistic_id

[in] ENUM_STATISTICS sayımının istatistiksel parametresinin tanımlayıcısı.

Dönüş değeri

Sınama sonuçlarından istatistiksel parametrenin değeri.

Not

Fonksiyon OnTester() veya OnDeinit() içinde, sınayıcıda çağrılabilir. Diğer durumlarda sonuç tanımsızdır.

TesterStop

Test etme sırasında program operasyonu tamamlanma komutunu verir.

```
void  TesterStop();
```

Geri dönüş değeri

Geri dönüş değeri yok.

Not

The TesterStop() fonksiyonu bir test temsilcisi üzerinde bir uzman danışmanın rutin erken kapanması için tasarlanmıştır - örneğin, belirli bir sayıdaki kayıp alım-satım işlemine veya önceden belirlenmiş bir zarar seviyesine ulaştığında.

TesterStop() çağrısı bir testin normal tamamlanması olarak kabul edilir, bu nedenle OnTester() fonksiyonu çağrılır ve birikmiş tüm alım-satım istatistikleri ve optimizasyon kriter değeri strateji sınayıcısına sunulur.

Strateji sınayıcısında ExpertRemove() un çağrılması da normal test tamamlanması anlamına gelir ve alım-satım istatistiklerini elde etmeyi sağlar, ancak burada uzman danışman temsilcinin hafızasından kaldırılır. Bu durumda, bir sonraki parametre setindeki geçişi tamamlamak için programı yeniden yüklemek zaman alacaktır. Bu nedenle, bir testin erken rutin tamamlanması için TesterStop() kullanılması tercih edilen bir seçenektir.

Ayrıca bakınız

Program Yürütme, Alım-Satım Stratejilerini Sınama, ExpertRemove, SetReturnError

TesterDeposit

Sınama sürecinde para yatırma işlemini taklit eden özel fonksiyon. Bazı para yönetim sistemlerinde kullanılabilir.

```
bool  TesterDeposit(
   double money      // yatırılacak toplam miktar
   );
```

Parametreler

money

[in] Yatırılan para biriminde hesaba yatırılacak para miktarı.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner.

Ayrıca bakınız

TesterWithdrawal

TesterWithdrawal

Sınama sürecinde para çekme işlemini taklit eden özel bir fonksiyon. Bazı varlık yönetim sistemlerinde kullanılabilir.

```
bool  TesterWithdrawal(
   double money      // çekilecek toplam
   );
```

Parametreler

money

[in] Çekmemiz gereken paranın toplamı (mevduat cinsinden).

Dönüş değeri

Başarılı sonuç durumunda true değerine, aksi durumda - false değerine dönüş yapar.

Ayrıca bakınız

TesterDeposit

TranslateKey

Mevcut giriş dili ve kontrol anahtarının durumuna göre, sanal tuş kodu ile bir Unicode karaktere dönüş yapar.

```
short  TranslateKey(
   int  key_code      // unicode karşılığı istenen tuş kodu
   );
```

Parametreler

key_code

[in] Tuş kodu.

Dönüş Değeri

Başarı durumunda, unicode karakter Hata durumunda -1 dönüşü yapar.

Not

Fonksiyon kullanıcının bastığı tuşu Unicode karaktere dönüştürmek için ToUnicodeEx kullanır. ToUnicodeEx tetiklenmezse hata oluşabilir - örneğin, SHIFT tuşunun karakterini almaya çalışırken.

Örnek:

```
void OnChartEvent(const int id,const long& lparam,const double& dparam,const string& s
  {
   if(id==CHARTEVENT_KEYDOWN)
     {
      short sym=TranslateKey((int)lparam);
      //--- girilen karakter başarıyla Unicode'a dönüştürülmüş ise
      if(sym>0)
         Print(sym,"'",ShortToString(sym),"'");
      else
         Print("TranslateKey hatası, tuş=",lparam);
     }
  }
```

Ayrıca bakınız

Müşteri Terminali Olayları, OnChartEvent

ZeroMemory

Referansla geçirilmiş bir değişkeni sıfırlar.

```
void  ZeroMemory(
   void & variable      // sıfırlama değişkeni
   );
```

Parametreler

variable

[in] [out] Sıfırlanmak istenen, referansla geçirilmiş değişken (sıfır değeri ile başlat).

Dönüş değeri

Dönüş değeri yok.

Not

Fonksiyon parametresinin bir dizgi olması çağrı değeri olarak NULL belirtmeye eşdeğerdir.
Basit tipler ve bunların dizileri için, aynı zamanda bu tipleri içeren yapılar/sınıflar için, bu basit bir sıfırlamadır.
Dizgiler ve dinamik diziler içeren nesnelerde, ZeroMemory() her eleman için çağrılır.
const modifier ile korunmayan diziler için bu, tüm elemanların sıfırlanmasıdır.
Karmaşık nesne dizilerinde, ZeroMemory() her eleman için çağrılır.

ZeroMemory(), korunan üyelere veya kalıtıma sahip sınıflara uygulanamaz.

Dizilerle Çalışmak için Kullanılan Fonksiyonlar Grubu

Diziler en fazla 4 boyutlu olabilir. Her boyut, 0 değerinden *boyut_sayısı-1* değerine kadar indislenir. Tek boyutlu ve 50 elemanlı bir diziyi örnek alırsak, ilk eleman dizi[0], son eleman ise dizi[49] şeklinde elde edilecektir.

Fonksiyon	Eylem
ArrayBsearch	Dizinin birinci boyutundaki birinci elemanın indis değerine dönüş yapar
ArrayCopy	Bir diziyi başka bir diziye kopyalar
ArrayCompare	Basit tipli veya karmaşık nesneler içermeyen özel yapıların karşılaştırma sonucunu verir
ArrayFree	Tüm dinamik dizilerin tamponunu (arabelleğini) boşaltır ve sıfır boyutunun eleman sayısını 0 yapar
ArrayGetAsSeries	Dizinin indisleme yönünü kontrol eder
ArrayInitialize	Nümerik dizinin tüm elemanlarını belirtilen değerle doldurur
ArrayFill	Diziyi belirtilen değerle doldurur
ArrayIsSeries	Dizinin bir zaman serisi olup olmadığını kontrol eder
ArrayIsDynamic	Dizinin dinamik olup olmadığını kontrol eder
ArrayMaximum	En büyük değerli elemanı arar
ArrayMinimum	En küçük değerli elemanı arar
ArrayPrint	Basit tipli bir diziyi veya bir yapıyı günlüğe yazar
ArrayRange	Dizinin belli bir boyutundaki eleman sayısına dönüş yapar
ArrayResize	Dizinin ilk boyutunun büyüklüğünü yeniden ayarlar
ArrayInsert	Belirtilen eleman sayısını belirli bir indeksten başlayarak bir kaynak diziden bir alıcı diziye ekler
ArrayRemove	Belirtilen eleman sayısını, belirtilen indeksten başlayarak diziden kaldırır
ArrayReverse	Dizideki belirtilen eleman sayısını, belirtilen indeksten başlayarak tersine çevirir
ArraySetAsSeries	Dizinin indisleme yönünü ayarlar
ArraySize	Dizideki eleman sayısına dönüş yapar
ArraySort	Nümerik dizileri ilk boyuta göre sıralar
ArraySwap	Aynı türdeki iki dinamik dizinin içeriğini değiştirir

ArrayBsearch

Artan şekilde sıralanmış çok-boyutlu bir sayısal dizi üzerinde, belirtilen değeri arar. Arama ilk boyuttaki elemanlar üzerinde gerçekleştirilir.

double tipli bir dizide arama yapmak için

```
int  ArrayBsearch(
   const double&    array[],   // aranacak dizi
   double           value      // ne aranıyor
   );
```

float tipli bir dizide arama yapmak için

```
int  ArrayBsearch(
   const float&    array[],   // aranacak dizi
   float           value      // ne için aranıyor
   );
```

long tipli bir dizide arama yapmak için

```
int  ArrayBsearch(
   const long&    array[],   // aranacak dizi
   long           value      // ne aranıyor
   );
```

int tipli bir dizide arama yapmak için

```
int  ArrayBsearch(
   const int&    array[],   // aranacak dizi
   int           value      // ne aranıyor
   );
```

short tipli bir dizide arama yapmak için

```
int  ArrayBsearch(
   const short&    array[],   // aranacak dizi
   short           value      // ne aranıyor
   );
```

char tipli bir dizide arama yapmak için

```
int  ArrayBsearch(
   const char&    array[],   // aranacak dizi
   char           value      // ne aranıyor
   );
```

Parametreler

array[]

[in] Aranacak sayısal dizi.

value

[in] Aranacak değer.

Dönüş değeri

Fonksiyon bulunan elemanın indisine dönüş yapar. Aranan eleman bulunamazsa, en yakın değerli elemanın indisine dönüş yapar.

Not

İkili aramalar sadece sıralanmış dizilerde işler. Sayısal dizileri sıralamak için ArraySort() fonksiyonunu kullanın.

Örnek:

```
#property description "RSI göstergesinin verilerine dayanan bu script"
#property description "belirlenen zaman aralığında, piyasadaki"
#property description "aşırı alım ve aşırı satım alanlarını göstermektedir."
//--- script çalıştırıldığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- giriş parametreleri
input int                InpMAPeriod=14;                  // Hareketli ortalama peri
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;     // Fiyat tipi
input double             InpOversoldValue=30.0;           // Aşırı satım seviyesi
input double             InpOverboughtValue=70.0;         // Aşırı alım seviyesi
input datetime           InpDateStart=D'2012.01.01 00:00'; // Analiz başlangıç tarihi
input datetime           InpDateFinish=D'2013.01.01 00:00'; // Analiz bitiş tarihi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   double rsi_buff[]; // gösterge değerlerinin dizisi
   int    size=0;     // dizi büyüklüğü
//--- RSI göstergesinin tanıtıcı değerini al
   ResetLastError();
   int rsi_handle=iRSI(Symbol(),Period(),InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- gösterge tanıtıcı değeri alınamadı
      PrintFormat("Gösterge tanıtıcı değerinin alınmasında hata. Hata kodu = %d",GetLa
      return;
     }
//--- gösterge, verileri hesaplayana kadar döngüde kalacak
   while(BarsCalculated(rsi_handle)==-1)
     {
      //--- eğer gösterge, betiğin işlemini zoraki tamamlamışsa çık
      if(IsStopped())
         return;
      //--- göstergenin değerlerini hesaplamasına izin vermek için kısa bir duraklama
      Sleep(10);
     }
```

```
//--- belli bir zaman aralığı için gösterge değerlerini kopyala
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,InpDateFinish,rsi_buff)==-1)
     {
      PrintFormat("Gösterge değerlerinin kopyalanması başarısız oldu. Hata kodu = %d",
      return;
     }
//--- dizi büyüklüğünü al
   size=ArraySize(rsi_buff);
//--- diziyi sırala
   ArraySort(rsi_buff);
//--- (yüzdelik terimlerle) piyasanın aşırı satım alanında olduğu zamanı bul
   double ovs=(double)ArrayBsearch(rsi_buff,InpOversoldValue)*100/(double)size;
//--- (yüzdelik terimlerle) piyasanın aşırı alım alanında olduğu zamanı bul
   double ovb=(double)(size-ArrayBsearch(rsi_buff,InpOverboughtValue))*100/(double)siz
//--- dizgilerden hareketle veriyi göstermek için
   string str=""+TimeToString(InpDateStart,TIME_DATE)+" tarihinden "
              +TimeToString(InpDateFinish,TIME_DATE)+" tarihine kadar piyasa:";
   string str_ovb="zamanın "+DoubleToString(ovb,2)+"% kadarında aşırı alım alanında";
   string str_ovs="zamanın "+DoubleToString(ovs,2)+"% kadarında aşırı satım alanında";
//--- veriyi çizelge üzerinde göster
   CreateLabel("top",5,60,str,clrDodgerBlue);
   CreateLabel("overbought",5,35,str_ovb,clrDodgerBlue);
   CreateLabel("oversold",5,10,str_ovs,clrDodgerBlue);
//--- çizelgeyi yeniden çiz
   ChartRedraw(0);
//--- durakla
   Sleep(10000);
  }
//+------------------------------------------------------------------+
//| Çizelgenin sol alt köşesinde yorumu göster                       |
//+------------------------------------------------------------------+
void CreateLabel(const string name,const int x,const int y,
                 const string str,const color clr)
  {
//--- etiketi oluştur
   ObjectCreate(0,name,OBJ_LABEL,0,0,0);
//--- etiketi sol alt köşeye bağla
   ObjectSetInteger(0,name,OBJPROP_CORNER,CORNER_LEFT_LOWER);
//--- tutturma konumunu değiştir
   ObjectSetInteger(0,name,OBJPROP_ANCHOR,ANCHOR_LEFT_LOWER);
//--- tutturma noktasının X yönünde uzaklığı
   ObjectSetInteger(0,name,OBJPROP_XDISTANCE,x);
//--- tutturma noktasının Y yönünde uzaklığı
   ObjectSetInteger(0,name,OBJPROP_YDISTANCE,y);
//--- etiket metni
   ObjectSetString(0,name,OBJPROP_TEXT,str);
//--- metin rengi
   ObjectSetInteger(0,name,OBJPROP_COLOR,clr);
```

```
//--- metin boyutu
   ObjectSetInteger(0,name,OBJPROP_FONTSIZE,12);
  }
```

ArrayCopy

Bir diziyi diğerine kopyalar.

```
int  ArrayCopy(
   void&        dst_array[],          // hedef dizi
   const void&  src_array[],          // kaynak dizi
   int          dst_start=0,          // hangi indisten başlanarak hedef diziye yazıla
   int          src_start=0,          // kaynak dizinin ilk indisi
   int          count=WHOLE_ARRAY     // eleman sayısı
   );
```

Parametreler

dst_array[]

[out] Hedef dizi

src_array[]

[in] Kaynak dizi

dst_start=0

[in] Hedef dizideki başlangıç indisi. Varsayılan olarak başlangıç indisi sıfırdır.

src_start=0

[in] Kaynak dizideki başlangıç indisi. Varsayılan olarak başlangıç indisi sıfırdır.

count=WHOLE_ARRAY

[in] Kopyalanması gereken eleman sayısı. Varsayılan olarak bütün dizi kopyalanır (count=WHOLE_ARRAY).

Dönüş değeri

Kopyalanan eleman sayısına dönüş yapar.

Not

'count<0' veya 'count>src_size-src_start' koşulları mevcutsa, dizinin kalan bölümlerinin tamamı kopyalanır. Diziler soldan sağa doğru kopyalanır. Serilerde ve dizilerde başlangıç pozisyonu soldan sağa kopyalama için tanımlanmış ve düzenlenmiştir.

Eğer diziler farklı tiptelerse, kopyalama sırasında kaynak dizinin her elemanının hedef dizinin tipine dönüştürülmesi denenecektir. Dizgi dizileri sadece dizgi dizilerine kopyalanabilir. Başlatma gerektiren nesneleri içeren sınıflardan ve yapılardan oluşan diziler kopyalanmazlar. Yapılardan oluşan bir dizi, sadece aynı tipli bir diziye kopyalanabilir.

İndisleme yönü zaman serilerindeki gibi olan dinamik diziler için, hedef dizinin büyüklüğü otomatik olarak kopyalanan veri sayısına genişletilir. Hedef dizinin büyüklüğü hiçbir durumda azaltılmaz.

Örnek:

```
#property description "Bu gösterge, yerel olarak yüksek ve düşük"
#property description "mumları vurgular. Uç değerlerin bulunacağı zaman aralığının"
#property description "uzunluğu bir giriş parametresiyle belirlenmelidir."
//--- gösterge ayarları
```

```
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//---- grafik
#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
//--- ön tanımlı sabitler
#define INDICATOR_EMPTY_VALUE 0.0
//--- giriş parametreleri
input int InpNum=4; // Yarım aralık uzunluğu
//--- gösterge tamponları
double ExtOpen[];
double ExtHigh[];
double ExtLow[];
double ExtClose[];
double ExtColor[];
//--- global değişkenler
int    ExtStart=0; // uçdeğer olmayan ilk mumun indisi
int    ExtCount=0; // uçdeğer olmayanların belirlenen aralıktaki sayısı
//+------------------------------------------------------------------+
//| Uçdeğer olmayan mumların doldurulması                            |
//+------------------------------------------------------------------+
void FillCandles(const double &open[],const double &high[],
                 const double &low[],const double &close[])
  {
//--- mumları doldur
   ArrayCopy(ExtOpen,open,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtHigh,high,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtLow,low,ExtStart,ExtStart,ExtCount);
   ArrayCopy(ExtClose,close,ExtStart,ExtStart,ExtCount);
  }
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,ExtOpen);
   SetIndexBuffer(1,ExtHigh);
   SetIndexBuffer(2,ExtLow);
   SetIndexBuffer(3,ExtClose);
   SetIndexBuffer(4,ExtColor,INDICATOR_COLOR_INDEX);
//--- gösterilmeyen değeri belirle
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- veri penceresinde gösterilebilmeleri için gösterge tamponlarını isimlendir
   PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
```

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- indisleme yönünü zaman serisi şeklinde ayarla
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
//--- çubukların hesaplanmasında kullanılan başlangıç değişkeni
   int start=prev_calculated;
//--- ilk InpNum*2 çubuk boyunca hesaplama başlamayacak
   if(start==0)
     {
      start+=InpNum*2;
      ExtStart=0;
      ExtCount=0;
     }
//--- çubuk şekil almışsa, bir sonraki potansiyel uçdeğeri kontrol et
   if(rates_total-start==1)
      start--;
//--- uçdeğer için kontrol edilecek çubuk indisi
   int ext;
//--- gösterge değeri hesaplama döngüsü
   for(int i=start;i<rates_total-1;i++)
     {
      //--- ilk olarak i indisli çubukta, çizim yapılmadan
      ExtOpen[i]=0;
      ExtHigh[i]=0;
      ExtLow[i]=0;
      ExtClose[i]=0;
      //--- kontrol amaçlı uçdeğer indisi
      ext=i-InpNum;
      //--- yerel maksimumu kontrol et
      if(IsMax(high,ext))
        {
```

```
         //--- uçdeğerli mumu vurgula
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=1;
         //--- uçdeğere kadar olan diğer çubukları belirsiz bir renk ile vurgula
         FillCandles(open,high,low,close);
         //--- değişken olan renkleri değiştir
         ExtStart=ext+1;
         ExtCount=0;
         //--- bir sonraki tekrara geç
         continue;
        }
      //--- yerel minimum için kontrol et
      if(IsMin(low,ext))
        {
         //--- uçdeğerli mumu vurgula
         ExtOpen[ext]=open[ext];
         ExtHigh[ext]=high[ext];
         ExtLow[ext]=low[ext];
         ExtClose[ext]=close[ext];
         ExtColor[ext]=2;
         //--- uçdeğere kadar olan diğer çubukları belirsiz bir renk ile vurgula
         FillCandles(open,high,low,close);
         //--- değişken olan değerleri değiştir
         ExtStart=ext+1;
         ExtCount=0;
         //--- bir sonraki tekrara geç
         continue;
        }
      //--- belirlenmiş aralıktaki uç olmayan değerlerin sayısını artır
      ExtCount++;
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Mevcut dizi elemanının yerel yüksek değer olup olmadığını kontrol et |
//+------------------------------------------------------------------+
bool IsMax(const double &price[],const int ind)
  {
//--- aralık başlangıç değişkeni
   int i=ind-InpNum;
//--- aralığın son periyotu
   int finish=ind+InpNum+1;
//--- aralığın ilk yarısını kontrol et
   for(;i<ind;i++)
     {
```

```
      if(price[ind]<=price[i])
         return(false);
      }
//--- aralığın ikinci yarısını kontrol et
   for(i=ind+1;i<finish;i++)
     {
      if(price[ind]<=price[i])
         return(false);
      }
//--- bu bir uçdeğer
   return(true);
  }
//+------------------------------------------------------------------+
//| Mevcut dizi elemanının yerel düşük değer olup olmadığını kontrol et |
//+------------------------------------------------------------------+
bool IsMin(const double &price[],const int ind)
  {
//--- aralık başlangıç değişkeni
   int i=ind-InpNum;
//--- aralık son değişkeni
   int finish=ind+InpNum+1;
//--- aralığın ilk yarısını kontrol et
   for(;i<ind;i++)
     {
      if(price[ind]>=price[i])
         return(false);
      }
//--- aralığın ikinci yarısını kontrol et
   for(i=ind+1;i<finish;i++)
     {
      if(price[ind]>=price[i])
         return(false);
      }
//--- bu bir uçdeğer
   return(true);
  }
```

ArrayCompare

Aynı tipteki iki dizinin karşılaştırma sonucuna dönüş yapar. Basit tipli veya karmaşık nesneler içermeyen özel yapılı (dizgiler, dinamik diziler, sınıflar ve karmaşık nesneli diğer yapıları içermeyen) dizilerin karşılaştırılmasında kullanılabilir.

```
int  ArrayCompare(
   const void&  array1[],           // birinci dizi
   const void&  array2[],           // ikinci dizi
   int          start1=0,           // ilk dizideki başlangıç konumu
   int          start2=0,           // ikinci dizideki başlangıç konumu
   int          count=WHOLE_ARRAY   // karşılaştırmada kullanılacak eleman sayısı
   );
```

Parametreler

array1[]

[in] Birinci dizi.

array2[]

[in] İkinci dizi.

start1=0

[in] Birinci dizide karşılaştırmanın başlayacağı ilk indis. Ön tanımlı başlangıç indisi - 0.

start2=0

[in] İkinci dizide karşılaştırmanın başlayacağı ilk indis. Ön tanımlı başlangıç indisi - 0.

count=WHOLE_ARRAY

[in] Karşılaştırılacak elemanların sayısı. Varsayılan olarak iki dizinin de tüm elemanları karşılaştırmaya katılır (count=WHOLE_ARRAY).

Dönüş değeri

- -1, array1[] değeri array2[] değerinden küçükse
- 0, array1[] değeri array2[] değerine eşitse
- 1, array1[] değeri array2[] değerinden büyükse
- -2, karşılaştırılan dizilerin tiplerinin uyumsuzluğu nedeniyle bir hata oluşursa veya start1[], start2[] ya da count değerleri dizinin dışına düşüyorsa.

Not

Eğer bir dizi, diğerinin birebir altkümesi ise ve büyüklük ile count=WHOLE_ARRAY değerleri farklıysa, fonksiyon sıfır değerine dönüş yapmayacaktır (diziler eşit sayılmayacaklardır). Dizilerin büyüklük karşılaştırmalarının sonucu, array1[] değeri array2[] değerinden küçükse '-1', değil ise '1' dönüşü yapılır.

ArrayFree

Bir dinamik dizinin tamponunu (arabelleğini) boşaltır ve sıfır boyutunun büyüklüğünü 0 olarak ayarlar.

```
void  ArrayFree(
   void&  array[]      // dizi
   );
```

Parametreler

array[]

[in] Dinamik dizi.

Dönüş değeri

Dönüş değeri yok.

Not

Kullanılan tüm hafızanın tek seferde boşaltıldığı ve dizilerle yapılan ana çalışmanın, gösterge tamponlarına erişimi de kapsadığı göz önüne alındığında, ArrayFree() fonksiyonunun kullanım gereksinimi çok sık görülmeyebilir. Tampon boyutları terminalin yönetim alt sistemi tarafından otomatik olarak yönetilir.

Uygulamanın kompleks dinamik ortamında kullanılan hafızanın kullanıcı tarafından yönetilmesi gerektiğinde, ArrayFree() fonksiyonu kullanıcının gereksiz dizilerle işgal edilmiş hafızayı anında ve net şekilde boşaltmasını sağlar.

Örnek:

```
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
#include <Controls\Label.mqh>
#include <Controls\ComboBox.mqh>
//--- ön tanımlı sabitler
#define X_START 0
#define Y_START 0
#define X_SIZE 280
#define Y_SIZE 300
//+------------------------------------------------------------------+
//| Hafıza ile çalışma amaçlı diyalog sınıfı                         |
//+------------------------------------------------------------------+
class CMemoryControl : public CAppDialog
  {
private:
   //--- dizi büyüklüğü
   int               m_arr_size;
   //--- diziler
   char              m_arr_char[];
   int               m_arr_int[];
   float             m_arr_float[];
   double            m_arr_double[];
```

```
   long              m_arr_long[];
   //--- etiketler
   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;
   //--- düğmeler
   CButton           m_button_add;
   CButton           m_button_free;
   //--- karma kutular
   CComboBox         m_combo_box_step;
   CComboBox         m_combo_box_type;
   //--- karma kutulardan dizi tipinin mevcut değeri
   int               m_combo_box_type_value;

public:
                     CMemoryControl(void);
                    ~CMemoryControl(void);
   //--- sınıf nesnesi oluşturma yöntemi
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- çizelge olayları işleyicisi
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- etiketleri oluştur
   bool              CreateLabel(CLabel &lbl,const string name,const int x,const int y
   //--- kontrol elemanlarını oluştur
   bool              CreateButton(CButton &button,const string name,const int x,const
   bool              CreateComboBoxStep(void);
   bool              CreateComboBoxType(void);
   //--- olay işleyicileri
   void              OnClickButtonAdd(void);
   void              OnClickButtonFree(void);
   void              OnChangeComboBoxType(void);
   //--- mevcut dizi ile çalışma yöntemi
   void              CurrentArrayFree(void);
   bool              CurrentArrayAdd(void);
  };
//+------------------------------------------------------------------+
//| Mevcut dizinin serbest belleği                                   |
//+------------------------------------------------------------------+
void CMemoryControl::CurrentArrayFree(void)
  {
//--- dizi büyüklüğünü sıfırla
```

```
   m_arr_size=0;
//--- diziyi serbest bırak
   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);
  }
//+------------------------------------------------------------------+
//| Mevcut dizi için bellek eklemeyi dene                            |
//+------------------------------------------------------------------+
bool CMemoryControl::CurrentArrayAdd(void)
  {
//--- kullanılan bellek fiziksel belleği aşarsa çık
   if(TerminalInfoInteger(TERMINAL_MEMORY_PHYSICAL)/TerminalInfoInteger(TERMINAL_MEMOR
      return(false);
//--- belleği mevcut tipe göre dağıtmayı dene
   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);
//--- bellek dağıtıldı
   return(true);
  }
//+------------------------------------------------------------------+
//| Olayların işlenmesi                                              |
//+------------------------------------------------------------------+
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)
//+------------------------------------------------------------------+
//| Yapıcı                                                           |
//+------------------------------------------------------------------+
CMemoryControl::CMemoryControl(void)
  {
  }
```

```
//+------------------------------------------------------------------+
//| Yıkıcı                                                           |
//+------------------------------------------------------------------+
CMemoryControl::~CMemoryControl(void)
  {
  }
//+------------------------------------------------------------------+
//| Sınıf nesnesi oluşturma yöntemi                                  |
//+------------------------------------------------------------------+
bool CMemoryControl::Create(const long chart,const string name,const int subwin,
                            const int x1,const int y1,const int x2,const int y2)
  {
//--- temel sınıf nesnesini oluştur
   if(!CAppDialog::Create(chart,name,subwin,x1,y1,x2,y2))
      return(false);
//--- dizgileri etiketler için hazırla
   string str_physical="Fiziksel hafıza = "+(string)TerminalInfoInteger(TERMINAL_MEMOR
   string str_total="Toplam hafıza = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_TOT
   string str_available="Mevcut hafıza = "+(string)TerminalInfoInteger(TERMINAL_MEMORY
   string str_used="Kullanılan hafıza = "+(string)TerminalInfoInteger(TERMINAL_MEMORY_
//--- etiketleri oluştur
   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,"Dizi tipi = "
      return(false);
   if(!CreateLabel(m_lbl_array_size,"size_label",X_START+10,Y_START+130,"Dizi büyüklüğ
      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,"Tipi
      return(false);
   if(!CreateLabel(m_lbl_add_size,"add_size_label",X_START+10,Y_START+210,"Diziye ekle
      return(false);
//--- kontrol elemanlarını oluştur
   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);
//--- değişkeni başlat
```

```
   m_arr_size=0;
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Düğmeyi oluştur                                                  |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateButton(CButton &button,const string name,const int x,
                                  const int y,const string str,const int font_size,
                                  const int clr)
  {
//--- düğmeyi oluştur
   if(!button.Create(m_chart_id,name,m_subwin,x,y,x+50,y+20))
      return(false);
//--- metin
   if(!button.Text(str))
      return(false);
//--- font boyutu
   if(!button.FontSize(font_size))
      return(false);
//--- etiket rengi
   if(!button.Color(clr))
      return(false);
//--- düğmeyi kontrol elemanlarına ekle
   if(!Add(button))
      return(false);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Dizi büyüklüğü için bir karma kutu oluştur                       |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxStep(void)
  {
//--- karma kutuyu oluştur
   if(!m_combo_box_step.Create(m_chart_id,"step_combobox",m_subwin,X_START+100,Y_START
      return(false);
//--- elemanları karma kutuya ekle
   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);
//--- mevcut karma kutu elemanını ayarla
   if(!m_combo_box_step.SelectByValue(1000000))
      return(false);
```

```
//--- karma kutuyu kontrol elemanlarına ekle
   if(!Add(m_combo_box_step))
      return(false);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Dizi tipi için bir karma kutu oluştur                            |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateComboBoxType(void)
  {
//--- karma kutuyu oluştur
   if(!m_combo_box_type.Create(m_chart_id,"type_combobox",m_subwin,X_START+100,Y_START
      return(false);
//--- elemanları karma kutuya ekle
   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);
//--- mevcut karma kutu elemanını ayarla
   if(!m_combo_box_type.SelectByValue(3))
      return(false);
//--- mevcut durumdaki karma kutu elemanını kaydet
   m_combo_box_type_value=3;
//--- karma kutuyu kontrol elemanlarına ekle
   if(!Add(m_combo_box_type))
      return(false);
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Bir etiket oluştur                                               |
//+------------------------------------------------------------------+
bool CMemoryControl::CreateLabel(CLabel &lbl,const string name,const int x,
                                 const int y,const string str,const int font_size,
                                 const int clr)
  {
//--- bir etiket oluştur
   if(!lbl.Create(m_chart_id,name,m_subwin,x,y,0,0))
      return(false);
//--- metin
   if(!lbl.Text(str))
      return(false);
```

```
//--- font boyutu
   if(!lbl.FontSize(font_size))
      return(false);
//--- renk
   if(!lbl.Color(clr))
      return(false);
//--- etiketi kontrol elemanlarına ekle
   if(!Add(lbl))
      return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| "Ekle" butonuna tıklama olayı işleyicisi                         |
//+------------------------------------------------------------------+
void CMemoryControl::OnClickButtonAdd(void)
  {
//--- dizi büyüklüğünü artır
   m_arr_size+=(int)m_combo_box_step.Value();
//--- mevcut dizi için belleği tahsis etmeyi dene
   if(CurrentArrayAdd())
     {
      //--- hafıza tahsis edildi, mevcut durumu ekranda göster
      m_lbl_memory_available.Text("Mevcut hafıza = "+(string)TerminalInfoInteger(TERMI
      m_lbl_memory_used.Text("Kullanılan hafıza = "+(string)TerminalInfoInteger(TERMIN
      m_lbl_array_size.Text("Dizi büyüklüğü = "+IntegerToString(m_arr_size));
      m_lbl_error.Text("");
     }
   else
     {
      //--- hafıza tahsisi başarısız oldu, hata mesajı göster
      m_lbl_error.Text("Dizi çok büyük, hata!");
      //--- önceki dizi büyüklüğüne dön
      m_arr_size-=(int)m_combo_box_step.Value();
     }
  }
//+------------------------------------------------------------------+
//| "Boşalt" düğmesinin tıklanması olayının işleyicisi               |
//+------------------------------------------------------------------+
void CMemoryControl::OnClickButtonFree(void)
  {
//--- mevcut dizinin belleğini boşalt
   CurrentArrayFree();
//--- mevcut durumu ekranda göster
   m_lbl_memory_available.Text("Mevcut hafıza = "+(string)TerminalInfoInteger(TERMINAI
   m_lbl_memory_used.Text("Kullanılan hafıza = "+(string)TerminalInfoInteger(TERMINAL_
   m_lbl_array_size.Text("Dizi büyüklüğü = 0");
   m_lbl_error.Text("");
  }
```

```
//+------------------------------------------------------------------+
//| Karma kutu değişimi olayının işleyicisi                          |
//+------------------------------------------------------------------+
void CMemoryControl::OnChangeComboBoxType(void)
  {
//--- dizi tipi değişmiş mi kontrol et
   if(m_combo_box_type.Value()!=m_combo_box_type_value)
     {
      //--- mevcut dizinin belleğini boşalt
      OnClickButtonFree();
      //--- başka bir dizi tipi ile çalış
      m_combo_box_type_value=(int)m_combo_box_type.Value();
      //--- yeni dizi tipini ekranda göster
      if(m_combo_box_type_value==0)
         m_lbl_array_type.Text("Dizi tipi = char");
      if(m_combo_box_type_value==1)
         m_lbl_array_type.Text("Dizi tipi = int");
      if(m_combo_box_type_value==2)
         m_lbl_array_type.Text("Dizi tipi = float");
      if(m_combo_box_type_value==3)
         m_lbl_array_type.Text("Dizi tipi = double");
      if(m_combo_box_type_value==4)
         m_lbl_array_type.Text("Dizi tipi = long");
     }
  }
//--- CMemoryControl sınıf nesnesi
CMemoryControl ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- diyaloğu oluştur
   if(!ExtDialog.Create(0,"MemoryControl",0,X_START,Y_START,X_SIZE,Y_SIZE))
      return(INIT_FAILED);
//--- çalıştır
   ExtDialog.Run();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
```

```
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

ArrayGetAsSeries

Dizi indisinin yönünü kontrol eder.

```
bool  ArrayGetAsSeries(
   const void&  array[]    // kontrol edilecek dizi
   );
```

Parametreler

array

[in] Kontrol edilen dizi.

Dönüş değeri

Eğer belirlenen dizi AS_SERIES bayrak ayarına sahipse (yani diziye erişim zaman serilerinde olduğu gibi arkadan öne doğru işliyorsa) true değerine dönüş yapar. Bir zaman serisi, elemanlarının indisleme yönünün sondan başa (en yeni veriden en eski veriye) doğru işlemesi açısından alışıldık diğer dizilerden ayrılır.

Not

Bir dizinin zaman serisi olup olmadığını kontrol etmek için ArrayIsSeries() fonksiyonunu kullanın. OnCalculate() fonksiyonuna giriş parametresi şeklinde geçirilen fiyat verisi dizileri, zaman serilerinde kullanılan indisleme yönüne sahip olmak zorunda değildir. Gereken indisleme yönü ArraySetAsSeries() fonksiyonu ile ayarlanabilir.

Örnek:

```
#property description "Gösterge, Açılış ve Kapanış fiyatları veya Yüksek ve Düşük fiya
#property description "arasındaki farkın mutlak değerini hesaplar ve bunları ayrı bir
#property description "bir histogram şeklinde gösterir."
//--- gösterge ayarları
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---- grafik
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_style1  STYLE_SOLID
#property indicator_width1  3
//--- giriş parametreleri
input bool InpAsSeries=true; // gösterge tamponunun indisleme yönü
input bool InpPrices=true;   // Hesaplama fiyatları (true - Open,Close; false - High,L
//--- gösterge tamponu
double ExtBuffer[];
//+------------------------------------------------------------------+
//| Gösterge değerlerinin hesaplanması                               |
//+------------------------------------------------------------------+
void CandleSizeOnBuffer(const int rates_total,const int prev_calculated,
                        const double &first[],const double &second[],double &buffer[])
  {
//--- çubukların hesaplanması için başlangıç değeri
```

```
   int start=prev_calculated;
//--- gösterge değerleri bir önceki tik ile zaten hesaplanmışsa son çubukta çalış
   if(prev_calculated>0)
      start--;
//--- dizilerin indisleme yönünü ayarla
   bool as_series_first=ArrayGetAsSeries(first);
   bool as_series_second=ArrayGetAsSeries(second);
   bool as_series_buffer=ArrayGetAsSeries(buffer);
//--- gerekirse indisleme yönünü doğrudan biriyle değiştir
   if(as_series_first)
      ArraySetAsSeries(first,false);
   if(as_series_second)
      ArraySetAsSeries(second,false);
   if(as_series_buffer)
      ArraySetAsSeries(buffer,false);
//--- gösterge değerlerini hesapla
   for(int i=start;i<rates_total;i++)
      buffer[i]=MathAbs(first[i]-second[i]);
  }
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarını bağla
   SetIndexBuffer(0,ExtBuffer);
//--- gösterge tamponunda indisleme elemanını ayarla
   ArraySetAsSeries(ExtBuffer,InpAsSeries);
//--- göstergenin hangi değerler için hesaplanacağını kontrol et
   if(InpPrices)
     {
      //--- Open ve Close fiyatları
      PlotIndexSetString(0,PLOT_LABEL,"BodySize");
      //--- gösterge rengini ayarla
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrOrange);
     }
   else
     {
      //--- Yüksek ve Düşük fiyatlar
      PlotIndexSetString(0,PLOT_LABEL,"ShadowSize");
      //--- gösterge rengini ayarla
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,clrDodgerBlue);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
```

```
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[])
  {
//--- göstergeyi bayrak değerine göre hesapla
   if(InpPrices)
      CandleSizeOnBuffer(rates_total,prev_calculated,open,close,ExtBuffer);
   else
      CandleSizeOnBuffer(rates_total,prev_calculated,high,low,ExtBuffer);
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Ayrıca Bakınız

Zaman Serilerine Erişim, ArraySetAsSeries

ArrayInitialize

Önceden ayarlanmış bir değerle, bir sayısal dizi başlatır.

For initialization of an array of char type

```
int  ArrayInitialize(
   char    array[],     // başlatılan dizi
   char    value        // ayarlanacak değer
   );
```

For initialization of an array of short type

```
int  ArrayInitialize(
   short   array[],     // başlatılan dizi
   short   value        // ayarlanacak değer
   );
```

For initialization of an array of int type

```
int  ArrayInitialize(
   int     array[],     // başlatılan dizi
   int     value        // ayarlanacak değer
   );
```

For initialization of an array of long type

```
int  ArrayInitialize(
   long    array[],     // başlatılan dizi
   long    value        // ayarlanacak değer
   );
```

For initialization of an array of float type

```
int  ArrayInitialize(
   float   array[],     // başlatılan dizi
   float   value        // ayarlanacak değer
   );
```

For initialization of an array of double type

```
int  ArrayInitialize(
   double  array[],     // başlatılan dizi
   double  value        // ayarlanacak değer
   );
```

For initialization of an array of bool type

```
int  ArrayInitialize(
   bool    array[],     // başlatılan dizi
   bool    value        // ayarlanacak değer
   );
```

For initialization of an array of uint type

```
int  ArrayInitialize(
   uint    array[],     // başlatılan dizi
   uint    value        // ayarlanacak değer
   );
```

Parametreler

array[]

[out] Başlatılması öngörülen sayısal dizi.

value

[in] Tüm dizi elemanlarının alacağı yeni değer.

Dönüş değeri

Elemanların sayısı.

Not

ArrayResize() fonksiyonu, sonradan yapılacak genişletmelerde hafızanın fiziksel yer değişimini önlemek için, dizi büyüklüğünün rezerve ara-bellek ile ayarlanmasını sağlar. Bu daha iyi bir performans elde etmek için uygulanır. Çünkü hafızanın yer değişim işlemi oldukça yavaştır.

Dizinin ArrayInitialize(array, init_val) kullanılarak başlatılması, dizi için ayrılmış rezerve elemanların sayısıyla başlatılacağı anlamına gelmez. ArrayResize() kullanılarak yapılan sonraki genişletmelerde elemanlar dizinin sonuna eklenecektir, değerleri tanımsız olacaktır ve çoğu durumda *init_value* değerine (başlangıç değerine) eşit olmayacaklardır.

Örnek:

```
void OnStart()
  {
//--- dinamik dizi
   double array[];
//--- dizi büyüklüğünü 100 elemana ayarlayalım ve fazladan 10 eleman için bir arabelle
   ArrayResize(array,100,10);
//--- dizi elemanlarını EMPTY_VALUE=DBL_MAX değeri (boş değer) ile başlat.
   ArrayInitialize(array,EMPTY_VALUE);
   Print("Başlatmanın ardından son 10 elemanın değerleri");
   for(int i=90;i<100;i++) printf("array[%d] = %G",i,array[i]);
//--- diziyi beş elemanla genişlet
   ArrayResize(array,105);
   Print("ArrayResize(array,105) çağrısının ardından son 10 elemanın değerleri ");
//--- son 5 elemanın değerleri rezerve edilmiş arabellekten alınır
   for(int i=95;i<105;i++) printf("array[%d] = %G",i,array[i]);
  }
```

ArrayFill

Diziyi belirtilen değerle doldurur.

```
void  ArrayFill(
   void&  array[],      // dizi
   int    start,         // başlangıç indisi
   int    count,         // doldurulacak eleman sayısı
   void   value          // değer
   );
```

Parametreler

array[]

[out] Basit tipli dizi (char, uchar, short, ushort, int, uint, long, ulong, bool, color, datetime, float, double).

start

[in] Başlangıç indisi. Benzer bir durumda AS_SERIES bayrağı gözardı edilir.

count

[in] Doldurulacak elemanların sayısı.

value

[in] Diziyi doldurmak için belirlenen değer.

Dönüş değeri

Dönüş değeri yok.

Not

ArrayFill() fonksiyonu çağrıldığında daima normal (soldan sağa) indisleme yönü kullanılır. Yani, ArraySetAsSeries() fonksiyonuyla dizi elemanlarının erişim sırası değiştirilse bile bu değişim gözardı edilecektir.

Çok boyutlu diziler, ArrayFill() fonksiyonunun kullanımı sırasında tek boyutlu gözükür. Örneğin, array[2][4] dizisi, array[8] olarak işlem görür. Yani, bu diziyle çalışırken başlangıç elemanının indisini 5 olacak şekilde belirlemelisiniz. Burada, array[2][4] dizisi için yapılan ArrayFill(array, 5, 2, 3.14) çağrısı, array[1][1] ve array[1][2] elemanlarını 3.14 değeri ile doldurur.

Örnek:

```
void OnStart()
  {
//--- dinamik dizi bildirimi
   int a[];
//--- büyüklüğü ayarla
   ArrayResize(a,10);
//--- ilk beş elemanı 123 ile doldur
   ArrayFill(a,0,5,123);
//--- sonraki beş elemanı 456 ile doldur
   ArrayFill(a,5,5,456);
```

```
//--- değerleri göster
   for(int i=0;i<ArraySize(a);i++) printf("a[%d] = %d",i,a[i]);
  }
```

ArrayIsDynamic

Dizinin dinamik olup olmadığını kontrol eder.

```
bool  ArrayIsDynamic(
   const void&  array[]    // kontrol edilen dizi
   );
```

Parametreler

array[]

[in] Kontrol edilen dizi.

Dönüş değeri

Seçilen dizi dinamik ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Örnek:

```
#property description "Bu gösterge değer hesaplamaz. ArrayFree() fonksiyon çağrısını;"
#property description "bir dinamik, bir statik ve bir göserge tamponu olmak üzere, üç
#property description "Sonuçlar Experts (Uzmanlar) bülteninde gösterilir."
//--- gösterge ayarları
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- global değişkenler
double ExtDynamic[];   // dinamik dizi
double ExtStatic[100]; // statik dizi
bool   ExtFlag=true;   // bayrak
double ExtBuff[];      // gösterge tamponu
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizi için bellek tahsis et
   ArrayResize(ExtDynamic,100);
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,ExtBuff);
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const int begin,
                const double &price[])
```

```
  {
//--- bir tekil analiz gerçekleştir
   if(ExtFlag)
     {
      //--- belleği diziler için boşaltmayı dene
      //--- 1. Dinamik dizi
      Print("+============================+");
      Print("1. Dinamik diziyi kontrol et:");
      Print("Belleğin boşalmasından önce büyüklük = ",ArraySize(ExtDynamic));
      Print("Bu bir dinamik dizi mi = ",ArrayIsDynamic(ExtDynamic) ? "Evet" : "Hayır");
      //--- dizi belleğini serbest bırakmayı dene
      ArrayFree(ExtDynamic);
      Print("Belleğin boşalmasından sonra büyüklük = ",ArraySize(ExtDynamic));
      //--- 2. Statik dizi
      Print("2. Statik diziyi kontrol et:");
      Print("Belleğin boşalmasından önce büyüklük = ",ArraySize(ExtStatic));
      Print("Bu bir dinamik dizi mi = ",ArrayIsDynamic(ExtStatic) ? "Evet" : "Hayır");
      //--- dizi belleğini serbest bırakmayı dene
      ArrayFree(ExtStatic);
      Print("Belleğin boşalmasından sonra büyüklük = ",ArraySize(ExtStatic));
      //--- 3. Gösterge tamponu
      Print("3. Gösterge tamponunu kontrol et:");
      Print("Belleğin boşalmasından önce büyüklük = ",ArraySize(ExtBuff));
      Print("Bu bir dinamik dizi mi = ",ArrayIsDynamic(ExtBuff) ? "Evet" : "Hayır");
      //--- dizi belleğini serbest bırakmayı dene
      ArrayFree(ExtBuff);
      Print("Belleğin boşalmasından sonra büyüklük = ",ArraySize(ExtBuff));
      //--- bayrak değerini değiştir
      ExtFlag=false;
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Ayrıca Bakınız

Zaman serileri ve göstergelere erişim

ArrayIsSeries

Bir dizinin zaman serisi olup olmadığını kontrol eder.

```
bool  ArrayIsSeries(
   const void&  array[]    // kontrol edilen dizi
   );
```

Parametreler

array[]

[in] Kontrol edilen dizi.

Dönüş değeri

Kontrol edilen dizinin zaman serisi olması durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapar. OnCalculate() fonksiyonuna parametre olarak geçirilen diziler, elemanlarının erişim sırası açısından ArrayGetAsSeries() aracılığıyla kontrol edilmelidir.

Örnek:

```
#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
//--- gösterge tamponları
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[] bir zaman serisi");
   else
      Print("open[] bir zaman serisi değil!!!");
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Ayrıca Bakınız

Zaman serileri ve göstergelere erişim

ArrayMaximum

Çok boyutlu bir sayısal dizinin ilk boyutundaki en büyük elemanı arar.

```
int  ArrayMaximum(
   const void&   array[],             // aranacak dizi
   int           start=0,             // incelemenin başlayacağı indis
   int           count=WHOLE_ARRAY    // incelenen eleman sayısı
   );
```

Parametreler

array[]

[in] Aramanın yapıldığı sayısal dizi.

start=0

[in] Kontrolün başlayacağı indis değeri.

count=WHOLE_ARRAY

[in] Aranacak eleman sayısı. Varsayılan olarak tüm diziyi arar (count=WHOLE_ARRAY).

Dönüş değeri

Fonksiyon, dizinin seriselliğini (indisleme yönünü) hesaba katarak, bulunan elemanın indis değerine dönüş yapar. Başarısızlık durumunda ise -1 değerine dönüş yapar.

Not

Maksimum değer aranırken AS_SERIES bayrağının değeri göz önünde bulundurulur.

ArrayMaximum ve ArrayMinimum fonksiyonları tüm dizileri parametre olarak kabul eder. Ama arama sadece ilk boyutta gerçekleştirilir.

Örnek:

```
#property description "Gösterge, daha büyük zaman dilimlerinin mumlarını şu andaki mum
//--- gösterge ayarları
#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
//--- ön tanımlı sabitler
#define INDICATOR_EMPTY_VALUE 0.0
//--- giriş parametreleri
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // Göstergenin hesaplanacağı z
input datetime        InpDateStart=D'2013.01.01 00:00'; // Analiz başlangıç tarihi
//--- satış eğilimli mumlar için oluşturulan gösterge tamponları
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- alım eğilimli mumlar için oluşturulan gösterge tamponları
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- global değişkenler
datetime ExtTimeBuff[];      // daha büyük zaman aralığı için zaman tamponu
int      ExtSize=0;          // zaman tamponu büyüklüğü
int      ExtCount=0;         // zaman tamponunu indisle
int      ExtStartPos=0;      // gösterge hesabı için başlangıç noktası
bool     ExtStartFlag=true;  // başlangıç noktasının belirlenmesi için yardımcı bayra
datetime ExtCurrentTime[1];  // daha büyük zaman aralığındaki son çubuğun oluşma zama
datetime ExtLastTime;        // daha büyük zaman aralığında, hesaplamanın gerçekleşti
bool     ExtBearFlag=true;   // alım eğilimli gösterge tamponlarına girilecek verinin
bool     ExtBullFlag=true;   // satış eğilimli gösterge tamponlarına girilecek verini
int      ExtIndexMax=0;      // dizideki en büyük değerli elemanın indisi
int      ExtIndexMin=0;      // dizideki en küçük değerli elemanın indisi
int      ExtDirectionFlag=0; // son mum için fiyat hareketi yönü
//--- doğru çizim için mumun açılış ve kapanış fiyatları arasında değişim yap
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
```

```
//+------------------------------------------------------------------+
//| Mumun temel parçasının doldurulması                              |
//+------------------------------------------------------------------+
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)
  {
//--- dizideki en büyük ve en küçük değerlerin indislerini bul
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // High dizisinin en büyüğü
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // Low dizisinin en küçüğü
//--- mevcut zaman aralığından kaç çubuk doldurulacak
   int count=fill_index-start+1;
//--- ilk çubuktaki kapanış değeri, son çubuktakini aşarsa mum satış yönlü olur
   if(open[start]>close[last])
     {
      //--- mumlar bunun öncesinde alım yönlüyse, alım yönlü gösterge tamponlarının de
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
      //--- satış yönlü mum
      ExtDirectionFlag=-1;
      //--- mumu oluştur
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     close[last],high[index_max],low[index_min],start,count,ExtBearFla
      //--- fonksiyondan çık
      return;
     }
//--- ilk çubuktaki kapanış fiyatı, son çubuğunkinden az ise mum, alım yönlü olur
   if(open[start]<close[last])
     {
      //--- mumlar bunun öncesinde satış yönlüyse, satış yönlü gösterge tamponlarının
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
      //--- alım yönlü mum
      ExtDirectionFlag=1;
      //--- mumu oluştur
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
                     open[start],high[index_max],low[index_min],start,count,ExtBullFla
      //--- fonksiyondan çıkış
      return;
     }
//--- eğer fonksiyonun bu bölümündeysen, ilk çubuktaki açılış fiyatı, son çubuktaki ka
//--- fiyatına eşitse mumun satış yönlü olduğu söylenir
//--- eğer mum bundan önce alım yönlü olmuşsa, alım yönlü gösterge tamponlarının değe
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowS
//--- satış yönlü mum
   ExtDirectionFlag=-1;
//--- eğer açılış ve kapanış değerleri eşitse, doğru gösterim için kaydırma kullan
```

```
   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);
  }
//+------------------------------------------------------------------+
//| Mumun sonunu doldur                                              |
//+------------------------------------------------------------------+
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)
  {
//--- tek çubuk varsa çizme
   if(last-start==0)
      return;
//--- ilk çubuktaki kapanış değeri, son çubuktakini aşarsa mum satış yönlü olur
   if(open[start]>close[last])
     {
      //--- mumun sonunu oluştur
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start],close[last],high[index_max],low[index_min],fill_index,
      //--- fonksiyondan çık
      return;
     }
//--- ilk çubuktaki kapanış fiyatı, son çubuğunkinden az ise mum, alım yönlü olur
   if(open[start]<close[last])
     {
      //--- mumun sonunu oluştur
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                    close[last],open[start],high[index_max],low[index_min],fill_index,
      //--- fonksiyondan çık
      return;
     }
//--- eğer fonksiyonun bu bölümündeysen, ilk çubuktaki açılış fiyatı, son çubuktaki ka
//--- fiyatına eşitse mumun satış yönlü olduğu söylenir
//--- mumun sonunu oluştur
   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
  }
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
```

```
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge periyotunu kontrol et
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- ön planda fiyat verisini göster
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- gösterge tamponlarının bağlanması
   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);
//--- göstergeyi oluşturmak için bazı özellikleri ayarla
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // grafiksel yapı tipi
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // çizgi stilinin çizimi
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // çizgi genişliğinin çizim
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
```

```
//--- halihazırda hesaplanan hiçbir çubuk olmaması durumunda
   if(prev_calculated==0)
     {
      //--- daha büyük zaman aralığındaki çubukların varış zamanını al
      if(!GetTimeData())
         return(0);
     }
//--- doğrudan indislemeyi ayarla
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- çubukların hesaplanması için başlangıç değeri
   int start=prev_calculated;
//--- eğer oluşmuşsa, göstergeyi çubuğun üzerinde hesapla
   if(start!=0 && start==rates_total)
      start--;
//--- gösterge değerlerinin hesaplanma döngüsü
   for(int i=start;i<rates_total;i++)
     {
      //--- gösterge tamponlarının i sayıda elemanını boş değerle doldur
      FillIndicatorBuffers(i);
      //--- hesaplamayı InpDateStart tarihinden başlayan çubuklar için uygula
      if(time[i]>=InpDateStart)
        {
         //--- değerlerin ilk olarak gösterilmeye başlanacağı pozisyonunu tanımla
         if(ExtStartFlag)
           {
            //--- başlangıç pozisyonunun numarasını kaydet
            ExtStartPos=i;
            //--- daha büyük zaman aralığında time[i] değerini aşan ilk tarihi tanımla
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- bir daha bu bloka rastlamamak için bayrak değerini değiştir
            ExtStartFlag=false;
           }
         //--- dizide hala elemanlar var mı kontrol et
         if(ExtCount<ExtSize)
           {
            //--- mevcut zaman aralığındaki değerin, daha büyük zaman aralığındaki değ
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- mumun ana kısmını çiz (son ve sondan bir önceki çubukların arasın
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,ExtI
               //--- mumun sonunu doldur (son ve sondan bir önceki çubukların arasında
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
               //--- bir sonraki mumu çizmek için başlangıç pozisyonunu kaydır
```

```
                ExtStartPos=i;
                //--- dizi sayacını artır
                ExtCount++;
               }
             else
                continue;
            }
          else
            {
             //--- dizi değerlerini sıfırla
             ResetLastError();
             //--- daha büyük zaman aralığından son tarihi al
             if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
               {
                Print("Veri kopyalama hatası, kod = ",GetLastError());
                return(0);
               }
             //--- yeni alınan tarih daha büyükse, mumun oluşturulmasını durdur
             if(ExtCurrentTime[0]>ExtLastTime)
               {
                //--- ana gösterge tamponlarında, son ve sondan bir önceki çubukların a
                ClearEndOfBodyMain(i-1);
                //--- yardımcı gösterge tamponlarını kullanarak alanı doldur
                FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
                //--- bir sonraki mumu çizmek için başlangıç pozisyonunu kaydır
                ExtStartPos=i;
                //--- fiyat yönü için oluşturulan bayrağı sıfırla
                ExtDirectionFlag=0;
                //--- yeni son tarihi kaydet
                ExtLastTime=ExtCurrentTime[0];
               }
             else
               {
                //--- mumu oluştur
                FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtIndex
               }
            }
         }
      }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Belirlenen gösterge periyotunun doğruluğunu kontrol et           |
//+------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
  {
//--- göstergenin periyotu, gösterildiği zaman aralığından büyük olmalı
   if(current_period>=high_period)
```

```
      {
       Print("Hata! Gösterge periyotunun değeri halihazırdaki zaman aralığı değerinden
       return(false);
      }
//--- gösterge periyotu bir hafta veya bir ay ise, periyot doğru
   if(high_period>32768)
       return(true);
//--- periyot değerlerini dakikalara dönüştür
   if(high_period>30)
       high_period=(high_period-16384)*60;
   if(current_period>30)
       current_period=(current_period-16384)*60;
//--- gösterge periyotu, üzerinde gösterildiği zaman aralığının katı olmalı
   if(high_period%current_period!=0)
     {
       Print("Hata! gösterge periyotunun değeri, üzerinde gösterildiği zaman aralığının
       return(false);
      }
//--- gösterge periyotu, üzerinde gösterildiği zaman aralığını en az 3 kat aşmalı
   if(high_period/current_period<3)
     {
       Print("Hata! Gösterge periyotu, üzerinde gösterildiği zaman aralığını en az 3 ka
       return(false);
      }
//--- halihazırdaki zaman aralığı için gösterge periyotu doğru
   return(true);
  }
//+------------------------------------------------------------------+
//| Daha büyük olan zaman aralığından zaman verisi al                |
//+------------------------------------------------------------------+
bool GetTimeData(void)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- mevcut zaman için tüm veriyi kopyala
   if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
     {
       //--- hata kodunu al
       int code=GetLastError();
       //--- hata mesajını çıktıla
       PrintFormat("Veri kopyalama hatası! %s",code==4401
                   ? "Geçmişin karşıya yüklenmesi hala devam ediyor!"
                   : "Kod = "+IntegerToString(code));
       //--- veri yükleme denemesini tekrarlamak için false değerine dönüş yap
       return(false);
      }
//--- dizi büyüklüğünü al
   ExtSize=ArraySize(ExtTimeBuff);
//--- dizi için oluşturulan döngü indisini sıfır yap
```

```
   ExtCount=0;
//--- şu andaki çubuğun zaman aralığı üzerindeki pozisyonunu sıfır yap
   ExtStartPos=0;
   ExtStartFlag=true;
//--- daha büyük zaman aralığındaki son zaman değerini kaydet
   ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Fonksiyon, mumun ana parçasını şekillendirir.    Bayrağın değerine       |
//| bağlı olarak, fonksiyon, hangi veri ve dizilerin doğru görüntüleme       |
//| amacıyla kullanılacağını belirler.                                       |
//+------------------------------------------------------------------------------+
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)
  {
//--- bayrağın değerini kontrol et
   if(flag)
     {
      //--- mumun gövdesini oluştur
      FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
      //--- mumun gölgesini oluştur
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- mumun gövdesini oluştur
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- mumun gölgesini oluştur
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------------------+
//| Fonksiyon, mumun sonunu şekillendirir.         Bayrak değerine bağlı olarak,  |
//| fonksiyon, hangi veri ve dizilerin düzgün görüntüleme                        |
//| amacıyla kullanılacağını belirler.                                           |
//+------------------------------------------------------------------------------+
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)
  {
//--- bayrağın değerini kontrol et
   if(flag)
```

```
      {
       //--- mum gövdesinin sonunu oluştur
       FormEnd(body_fst,body_snd,fst_value,snd_value,end);
       //--- mum gölgesinin sonunu oluştur
       FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
       //--- bayrak değerini tersine değiştir
       flag=false;
      }
    else
      {
       //--- mum gövdesinin sonunu oluştur
       FormEnd(body_fst,body_snd,snd_value,fst_value,end);
       //--- mum gölgesinin sonunu oluştur
       FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
       //--- bayrak değerini tersine değiştir
       flag=true;
      }
  }
//+------------------------------------------------------------------------------------+
//| Mumun son kısmını (bir önceki çubuk ile şimdiki çubuk arasında kalan alanı)      |
//| temizle)                                                                           |
//+------------------------------------------------------------------------------------+
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecc
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecc
  }
//+------------------------------------------------------------------------------+
//| Mumu temizle                                                                 |
//+------------------------------------------------------------------------------+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- kontrol et
   if(count!=0)
      {
       //--- gösterge tamponlarını boş değerlerle doldur
       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);
      }
  }
//+------------------------------------------------------------------------------+
//| Mumun ana parçasını oluştur                                                  |
//+------------------------------------------------------------------------------+
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
  {
```

```
//--- kontrol et
   if(count!=0)
     {
      //--- gösterge tamponlarını değerlerle doldur
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------------------+
//| Mumun sonunu oluştur                                                         |
//+------------------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- gösterge tamponlarını değerlerle doldur
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
  }
//+------------------------------------------------------------------+
//| Gösterge tamponlarının i elemanını boş değerle doldur            |
//+------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//--- gösterge tamponlarının hücresinde bir boş değer ayarla
   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

Çok boyutlu bir sayısal dizinin ilk boyutundaki en küçük elemanı arar.

```
int  ArrayMinimum(
   const void&   array[],             // aranacak dizi
   int           start=0,             // incelemenin başlayacağı indis
   int           count=WHOLE_ARRAY    // incelenen eleman sayısı
   );
```

Parametreler

array[]

[in] Aramanın yapıldığı sayısal dizi.

start=0

[in] Kontrolün başlayacağı indis değeri.

count=WHOLE_ARRAY

[in] Aranacak eleman sayısı. Varsayılan olarak tüm diziyi arar (count=WHOLE_ARRAY).

Dönüş değeri

Fonksiyon, dizinin seriselliğini (indisleme yönünü) hesaba katarak, bulunan elemanın indis değerine dönüş yapar. Başarısızlık durumunda ise -1 değerine dönüş yapar.

Not

Minimum değer aranırken AS_SERIES bayrağının değeri hesaba katılır.

ArrayMaximum ve ArrayMinimum fonksiyonları tüm dizileri parametre olarak kabul eder. Ama arama sadece ilk boyutta gerçekleştirilir.

Örnek:

```
#property description "Gösterge, daha büyük zaman dilimlerinin mumlarını şu andaki mum
//--- gösterge ayarları
#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
//--- ön tanımlı sabitler
#define INDICATOR_EMPTY_VALUE 0.0
//--- giriş parametreleri
input ENUM_TIMEFRAMES InpPeriod=PERIOD_H4;              // Göstergenin hesaplanacağı z
input datetime        InpDateStart=D'2013.01.01 00:00'; // Analiz başlangıç tarihi
//--- satış eğilimli mumlar için oluşturulan gösterge tamponları
double   ExtBearBodyFirst[];
double   ExtBearBodySecond[];
double   ExtBearBodyEndFirst[];
double   ExtBearBodyEndSecond[];
double   ExtBearShadowFirst[];
double   ExtBearShadowSecond[];
double   ExtBearShadowEndFirst[];
double   ExtBearShadowEndSecond[];
//--- alım eğilimli mumlar için oluşturulan gösterge tamponları
double   ExtBullBodyFirst[];
double   ExtBullBodySecond[];
double   ExtBullBodyEndFirst[];
double   ExtBullBodyEndSecond[];
double   ExtBullShadowFirst[];
double   ExtBullShadowSecond[];
double   ExtBullShadowEndFirst[];
double   ExtBullShadowEndSecond[];
//--- global değişkenler
datetime ExtTimeBuff[];      // daha büyük zaman aralığı için zaman tamponu
int      ExtSize=0;          // zaman tamponu büyüklüğü
int      ExtCount=0;         // zaman tamponunu indisle
int      ExtStartPos=0;      // gösterge hesabı için başlangıç noktası
bool     ExtStartFlag=true;  // başlangıç noktasının belirlenmesi için yardımcı bayra
datetime ExtCurrentTime[1];  // daha büyük zaman aralığındaki son çubuğun oluşma zama
datetime ExtLastTime;        // daha büyük zaman aralığında, hesaplamanın gerçekleşti
bool     ExtBearFlag=true;   // alım eğilimli gösterge tamponlarına girilecek verinin
bool     ExtBullFlag=true;   // satış eğilimli gösterge tamponlarına girilecek verini
int      ExtIndexMax=0;      // dizideki en büyük değerli elemanın indisi
int      ExtIndexMin=0;      // dizideki en küçük değerli elemanın indisi
int      ExtDirectionFlag=0; // son mum için fiyat hareketi yönü
//--- doğru çizim için mumun açılış ve kapanış fiyatları arasında değişim yap
const double ExtEmptyBodySize=0.2*SymbolInfoDouble(Symbol(),SYMBOL_POINT);
```

```
//+------------------------------------------------------------------+
//| Mumun temel parçasının doldurulması                              |
//+------------------------------------------------------------------+
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)
  {
//--- dizideki en büyük ve en küçük değerlerin indislerini bul
   index_max=ArrayMaximum(high,ExtStartPos,last-start+1); // High dizisinin en büyüğü
   index_min=ArrayMinimum(low,ExtStartPos,last-start+1);  // Low dizisinin en küçüğü
//--- mevcut zaman aralığından kaç çubuk doldurulacak
   int count=fill_index-start+1;
//--- ilk çubuktaki kapanış değeri, son çubuktakini aşarsa mum satış yönlü olur
   if(open[start]>close[last])
     {
      //--- mumlar bunun öncesinde alım yönlüyse, alım yönlü gösterge tamponlarının de
      if(ExtDirectionFlag!=-1)
         ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
      //--- satış yönlü mum
      ExtDirectionFlag=-1;
      //--- mumu oluştur
      FormCandleMain(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
                     close[last],high[index_max],low[index_min],start,count,ExtBearFla
      //--- fonksiyondan çık
      return;
     }
//--- ilk çubuktaki kapanış fiyatı, son çubuğunkinden az ise mum, alım yönlü olur
   if(open[start]<close[last])
     {
      //--- mumlar bunun öncesinde satış yönlüyse, satış yönlü gösterge tamponlarının
      if(ExtDirectionFlag!=1)
         ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShad
      //--- alım yönlü mum
      ExtDirectionFlag=1;
      //--- mumu oluştur
      FormCandleMain(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShad
                     open[start],high[index_max],low[index_min],start,count,ExtBullFla
      //--- fonksiyondan çıkış
      return;
     }
//--- eğer fonksiyonun bu bölümündeysen, ilk çubuktaki açılış fiyatı, son çubuktaki ka
//--- fiyatına eşitse mumun satış yönlü olduğu söylenir
//--- eğer mum bundan önce alım yönlü olmuşsa, alım yönlü gösterge tamponlarının değe
   if(ExtDirectionFlag!=-1)
      ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowS
//--- satış yönlü mum
   ExtDirectionFlag=-1;
//--- eğer açılış ve kapanış değerleri eşitse, doğru gösterim için kaydırma kullan
```

```
   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);
  }
//+------------------------------------------------------------------+
//| Mumun sonunu doldur                                              |
//+------------------------------------------------------------------+
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)
  {
//--- tek çubuk varsa çizme
   if(last-start==0)
      return;
//--- ilk çubuktaki kapanış değeri, son çubuktakini aşarsa mum satış yönlü olur
   if(open[start]>close[last])
     {
      //--- mumun sonunu oluştur
      FormCandleEnd(ExtBearBodyEndFirst,ExtBearBodyEndSecond,ExtBearShadowEndFirst,Ext
                    open[start],close[last],high[index_max],low[index_min],fill_index,
      //--- fonksiyondan çık
      return;
     }
//--- ilk çubuktaki kapanış fiyatı, son çubuğunkinden az ise mum, alım yönlü olur
   if(open[start]<close[last])
     {
      //--- mumun sonunu oluştur
      FormCandleEnd(ExtBullBodyEndFirst,ExtBullBodyEndSecond,ExtBullShadowEndFirst,Ext
                    close[last],open[start],high[index_max],low[index_min],fill_index,
      //--- fonksiyondan çık
      return;
     }
//--- eğer fonksiyonun bu bölümündeysen, ilk çubuktaki açılış fiyatı, son çubuktaki ka
//--- fiyatına eşitse mumun satış yönlü olduğu söylenir
//--- mumun sonunu oluştur
   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
  }
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
```

```
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge periyotunu kontrol et
   if(!CheckPeriod((int)Period(),(int)InpPeriod))
      return(INIT_PARAMETERS_INCORRECT);
//--- ön planda fiyat verisini göster
   ChartSetInteger(0,CHART_FOREGROUND,0,1);
//--- gösterge tamponlarının bağlanması
   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);
//--- göstergeyi oluşturmak için bazı özellikleri ayarla
   for(int i=0;i<8;i++)
     {
      PlotIndexSetInteger(i,PLOT_DRAW_TYPE,DRAW_FILLING); // grafiksel yapı tipi
      PlotIndexSetInteger(i,PLOT_LINE_STYLE,STYLE_SOLID); // çizgi stilinin çizimi
      PlotIndexSetInteger(i,PLOT_LINE_WIDTH,1);           // çizgi genişliğinin çizim
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
```

```
//--- halihazırda hesaplanan hiçbir çubuk olmaması durumunda
   if(prev_calculated==0)
     {
      //--- daha büyük zaman aralığındaki çubukların varış zamanını al
      if(!GetTimeData())
         return(0);
     }
//--- doğrudan indislemeyi ayarla
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(close,false);
//--- çubukların hesaplanması için başlangıç değeri
   int start=prev_calculated;
//--- eğer oluşmuşsa, göstergeyi çubuğun üzerinde hesapla
   if(start!=0 && start==rates_total)
      start--;
//--- gösterge değerlerinin hesaplanma döngüsü
   for(int i=start;i<rates_total;i++)
     {
      //--- gösterge tamponlarının i sayıda elemanını boş değerle doldur
      FillIndicatorBuffers(i);
      //--- hesaplamayı InpDateStart tarihinden başlayan çubuklar için uygula
      if(time[i]>=InpDateStart)
        {
         //--- değerlerin ilk olarak gösterilmeye başlanacağı pozisyonunu tanımla
         if(ExtStartFlag)
           {
            //--- başlangıç pozisyonunun numarasını kaydet
            ExtStartPos=i;
            //--- daha büyük zaman aralığında time[i] değerini aşan ilk tarihi tanımla
            while(time[i]>=ExtTimeBuff[ExtCount])
               if(ExtCount<ExtSize-1)
                  ExtCount++;
            //--- bir daha bu bloka rastlamamak için bayrak değerini değiştir
            ExtStartFlag=false;
           }
         //--- dizide hala elemanlar var mı kontrol et
         if(ExtCount<ExtSize)
           {
            //--- mevcut zaman aralığındaki değerin, daha büyük zaman aralığındaki değ
            if(time[i]>=ExtTimeBuff[ExtCount])
              {
               //--- mumun ana kısmını çiz (son ve sondan bir önceki çubukların arasın
               FillCandleMain(open,close,high,low,ExtStartPos,i-1,i-2,ExtIndexMax,ExtI
               //--- mumun sonunu doldur (son ve sondan bir önceki çubukların arasında
               FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
               //--- bir sonraki mumu çizmek için başlangıç pozisyonunu kaydır
```

```
                ExtStartPos=i;
                //--- dizi sayacını artır
                ExtCount++;
               }
             else
                continue;
            }
          else
            {
             //--- dizi değerlerini sıfırla
             ResetLastError();
             //--- daha büyük zaman aralığından son tarihi al
             if(CopyTime(Symbol(),InpPeriod,0,1,ExtCurrentTime)==-1)
               {
                Print("Veri kopyalama hatası, kod = ",GetLastError());
                return(0);
               }
             //--- yeni alınan tarih daha büyükse, mumun oluşturulmasını durdur
             if(ExtCurrentTime[0]>ExtLastTime)
               {
                //--- ana gösterge tamponlarında, son ve sondan bir önceki çubukların a
                ClearEndOfBodyMain(i-1);
                //--- yardımcı gösterge tamponlarını kullanarak alanı doldur
                FillCandleEnd(open,close,high,low,ExtStartPos,i-1,i-1,ExtIndexMax,ExtIn
                //--- bir sonraki mumu çizmek için başlangıç pozisyonunu kaydır
                ExtStartPos=i;
                //--- fiyat yönü için oluşturulan bayrağı sıfırla
                ExtDirectionFlag=0;
                //--- yeni son tarihi kaydet
                ExtLastTime=ExtCurrentTime[0];
               }
             else
               {
                //--- mumu oluştur
                FillCandleMain(open,close,high,low,ExtStartPos,i,i,ExtIndexMax,ExtIndex
               }
            }
         }
      }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Belirlenen gösterge periyotunun doğruluğunu kontrol et           |
//+------------------------------------------------------------------+
bool CheckPeriod(int current_period,int high_period)
  {
//--- göstergenin periyotu, gösterildiği zaman aralığından büyük olmalı
   if(current_period>=high_period)
```

```
      {
       Print("Hata! Gösterge periyotunun değeri halihazırdaki zaman aralığı değerinden
       return(false);
      }
//--- gösterge periyotu bir hafta veya bir ay ise, periyot doğru
   if(high_period>32768)
       return(true);
//--- periyot değerlerini dakikalara dönüştür
   if(high_period>30)
       high_period=(high_period-16384)*60;
   if(current_period>30)
       current_period=(current_period-16384)*60;
//--- gösterge periyotu, üzerinde gösterildiği zaman aralığının katı olmalı
   if(high_period%current_period!=0)
     {
       Print("Hata! gösterge periyotunun değeri, üzerinde gösterildiği zaman aralığının
       return(false);
      }
//--- gösterge periyotu, üzerinde gösterildiği zaman aralığını en az 3 kat aşmalı
   if(high_period/current_period<3)
     {
       Print("Hata! Gösterge periyotu, üzerinde gösterildiği zaman aralığını en az 3 ka
       return(false);
      }
//--- halihazırdaki zaman aralığı için gösterge periyotu doğru
   return(true);
  }
//+------------------------------------------------------------------+
//| Daha büyük olan zaman aralığından zaman verisi al                |
//+------------------------------------------------------------------+
bool GetTimeData(void)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- mevcut zaman için tüm veriyi kopyala
   if(CopyTime(Symbol(),InpPeriod,InpDateStart,TimeCurrent(),ExtTimeBuff)==-1)
     {
       //--- hata kodunu al
       int code=GetLastError();
       //--- hata mesajını çıktıla
       PrintFormat("Veri kopyalama hatası! %s",code==4401
                   ? "Geçmişin karşıya yüklenmesi hala devam ediyor!"
                   : "Kod = "+IntegerToString(code));
       //--- veri yükleme denemesini tekrarlamak için false değerine dönüş yap
       return(false);
      }
//--- dizi büyüklüğünü al
   ExtSize=ArraySize(ExtTimeBuff);
//--- dizi için oluşturulan döngü indisini sıfır yap
```

```
   ExtCount=0;
//--- şu andaki çubuğun zaman aralığı üzerindeki pozisyonunu sıfır yap
   ExtStartPos=0;
   ExtStartFlag=true;
//--- daha büyük zaman aralığındaki son zaman değerini kaydet
   ExtLastTime=ExtTimeBuff[ExtSize-1];
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------------------+
//| Fonksiyon, mumun ana parçasını şekillendirir.    Bayrağın değerine       |
//| bağlı olarak, fonksiyon, hangi veri ve dizilerin doğru görüntüleme       |
//| amacıyla kullanılacağını belirler.                                       |
//+------------------------------------------------------------------------------+
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)
  {
//--- bayrağın değerini kontrol et
   if(flag)
     {
      //--- mumun gövdesini oluştur
      FormMain(body_fst,body_snd,fst_value,snd_value,start,count);
      //--- mumun gölgesini oluştur
      FormMain(shadow_fst,shadow_snd,fst_extremum,snd_extremum,start,count);
     }
   else
     {
      //--- mumun gövdesini oluştur
      FormMain(body_fst,body_snd,snd_value,fst_value,start,count);
      //--- mumun gölgesini oluştur
      FormMain(shadow_fst,shadow_snd,snd_extremum,fst_extremum,start,count);
     }
  }
//+------------------------------------------------------------------------------+
//| Fonksiyon, mumun sonunu şekillendirir.         Bayrağın değerine bağlı olarak, |
//| fonksiyon, hangi veri ve dizilerin düzgün görüntüleme                    |
//| amacıyla kullanılacağını belirler.                                       |
//+------------------------------------------------------------------------------+
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)
  {
//--- bayrağın değerini kontrol et
   if(flag)
```

```
      {
       //--- mum gövdesinin sonunu oluştur
       FormEnd(body_fst,body_snd,fst_value,snd_value,end);
       //--- mum gölgesinin sonunu oluştur
       FormEnd(shadow_fst,shadow_snd,fst_extremum,snd_extremum,end);
       //--- bayrak değerini tersine değiştir
       flag=false;
      }
    else
      {
       //--- mum gövdesinin sonunu oluştur
       FormEnd(body_fst,body_snd,snd_value,fst_value,end);
       //--- mum gölgesinin sonunu oluştur
       FormEnd(shadow_fst,shadow_snd,snd_extremum,fst_extremum,end);
       //--- bayrak değerini tersine değiştir
       flag=true;
      }
  }
//+------------------------------------------------------------------------------------------+
//| Mumun son kısmını (bir önceki çubuk ile şimdiki çubuk arasında kalan alanı)
//| temizle)
//+------------------------------------------------------------------------------------------+
void ClearEndOfBodyMain(const int ind)
  {
   ClearCandle(ExtBearBodyFirst,ExtBearBodySecond,ExtBearShadowFirst,ExtBearShadowSecc
   ClearCandle(ExtBullBodyFirst,ExtBullBodySecond,ExtBullShadowFirst,ExtBullShadowSecc
  }
//+------------------------------------------------------------------+
//| Mumu temizle                                                     |
//+------------------------------------------------------------------+
void ClearCandle(double &body_fst[],double &body_snd[],double &shadow_fst[],
                 double &shadow_snd[],const int start,const int count)
  {
//--- kontrol et
   if(count!=0)
      {
       //--- gösterge tamponlarını boş değerlerle doldur
       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);
      }
  }
//+------------------------------------------------------------------+
//| Mumun ana parçasını oluştur                                      |
//+------------------------------------------------------------------+
void FormMain(double &fst[],double &snd[],const double fst_value,
              const double snd_value,const int start,const int count)
  {
```

```
//--- kontrol et
   if(count!=0)
     {
      //--- gösterge tamponlarını değerlerle doldur
      ArrayFill(fst,start,count,fst_value);
      ArrayFill(snd,start,count,snd_value);
     }
  }
//+------------------------------------------------------------------+
//| Mumun sonunu oluştur                                             |
//+------------------------------------------------------------------+
void FormEnd(double &fst[],double &snd[],const double fst_value,
             const double snd_value,const int last)
  {
//--- gösterge tamponlarını değerlerle doldur
   ArrayFill(fst,last-1,2,fst_value);
   ArrayFill(snd,last-1,2,snd_value);
  }
//+------------------------------------------------------------------+
//| Gösterge tamponlarının i elemanını boş değerle doldur            |
//+------------------------------------------------------------------+
void FillIndicatorBuffers(const int i)
  {
//--- gösterge tamponlarının hücresinde bir boş değer ayarla
   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

Basit tipli bir diziyi veya bir yapıyı günlüğe yazar.

```
void  ArrayPrint(
   const void&   array[],             // yazılacak dizi
   uint          digits=_Digits,      // ondalık basamakların sayısı
   const string  separator=NULL,      // yapı alanı değerlerinin ayracı
   ulong         start=0,             // yazılacak ilk elemanın indisi
   ulong         count=WHOLE_ARRAY,   // yazılacak elemanların sayısı
   ulong         flags=ARRAYPRINT_HEADER|ARRAYPRINT_INDEX|ARRAYPRINT_LIMIT|ARRAYPRINT_
   );
```

Parametreler

array[]

[in] Basit tipli bir dizi veya bir basit yapı.

digits=_Digits

[in] Reel tipler için ondalık basamak sayısı. Varsayılan değer olarak _Digits kullanılır.

separator=NULL

[in] Yapı elemanları alanının değerleri için kullanılan ayraç. Varsayılan değer NULL boş satır anlamına gelir. Bu durumda ayraç olarak boşluk karakteri kullanılır.

start=0

[in] Yazılacak ilk dizi elemanının indisi. Yazma işlemine sıfırıncı indisten başlanır.

count=WHOLE_ARRAY

[in] Yazılacak dizi elemanlarının sayısı. Varsayılan olarak tüm dizi yazılır (count=WHOLE_ARRAY).

flags=ARRAYPRINT_HEADER|ARRAYPRINT_INDEX|ARRAYPRINT_LIMIT|ARRAYPRINT_ALIGN

[in] Çıktı modunu ayarlayan bayrak kombinasyonu. Varsayılan olarak tüm bayraklar etkindir:

- ARRAYPRINT_HEADER - yapı dizisinin başlıklarını yaz
- ARRAYPRINT_INDEX - indisi sol tarafa yaz
- ARRAYPRINT_LIMIT - sadece ilk ve son 100 dizi elemanını yaz. Bunu sadece çok geniş bir diziyi yazdırırken kullanın.
- ARRAYPRINT_ALIGN - yazılan değerler için hizalamayı etkinleştir - sayılar sağa, metinler sola yaslanır.
- ARRAYPRINT_DATE - tarih/zaman yazılırken tarihi dd.mm.yyyy (gün.ay.yıl) formatında yaz
- ARRAYPRINT_MINUTES - tarih/zaman yazılırken, zamanı HH:MM (saat:dakika) formatında yaz
- ARRAYPRINT_SECONDS - tarih/zaman yazılırken, zamanı HH:MM:SS (saat:dakika:saniye) formatında yaz

Dönüş Değeri

Yok

Not

ArrayPrint() tüm yapı dizisi alanlarını günlüğe yazmaz - dizi ve nesne işaretçisi alanları atlanır. Daha uygun bir sunum için bu sütunlar yazılmaz. Tüm yapı alanlarını yazdırmanız gerekiyorsa, istediğiniz şekillendirme yöntemiyle kendi yazdırma fonksiyonunuzu oluşturmalısınız.

Örnek:

```
//--- son 10 çubuğun değerlerini yazdır
   MqlRates rates[];
   if(CopyRates(_Symbol,_Period,1,10,rates))
     {
      ArrayPrint(rates);
      Print("Denetle\n[zaman]\t[açılış]\t[yüksek]\t[düşük]\t[kapanış]\t[tik_hacmi]\t[r
      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 başarısız oldu, hata kodu=%d",GetLastError());
//--- çıktı örneği
/*
                    [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
  Denetle
  [zaman] [açılış] [yüksek] [düşük] [kapanış] [tik_hacmi] [makas] [reel_hacim]
   [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
*/
```

Ayrıca bakınız

FileSave, FileLoad

ArrayRange

Dizinin belirtilen boyutundaki elemanların sayısını verir.

```
int  ArrayRange(
   const void&   array[],      // kontrol edilecek dizi
   int           rank_index    // boyutun indisi
   );
```

Parametreler

array[]

[in] Kontrol edilen dizi.

rank_index

[in] Boyutun indisi.

Dönüş değeri

Seçilen dizi boyutundaki eleman sayısı.

Not

İndisler sıfırdan başladığı sürece, dizi boyutlarının sayısı son boyutun indis değerinden bir fazla olacaktır.

Örnek:

```
void OnStart()
  {
//--- dört boyutlu dizi oluştur
   double array[][5][2][4];
//--- sıfır boyutunun büyüklüğünü ayarla
   ArrayResize(array,10,10);
//--- boyutları yaz
   int temp;
   for(int i=0;i<4;i++)
     {
      //--- i boyutunun büyüklüğünü hesapla
      temp=ArrayRange(array,i);
      //--- yaz
      PrintFormat("boyut = %d, kapsam = %d",i,temp);
     }
//--- Sonuç
// boyut = 0, kapsam = 10
// boyut = 1, kapsam = 5
// boyut = 2, kapsam = 2
// boyut = 3, kapsam = 4
  }
```

ArrayResize

Dizinin ilk boyutunun eleman sayısını değiştirir

```
int  ArrayResize(
   void&  array[],              // referansla geçirilen dizi
   int    new_size,             // yeni eleman sayısı
   int    reserve_size=0        // rezerve büyüklük değeri (ilave)
   );
```

Parametreler

array[]

[out] Büyüklüğü değiştirilecek dizi.

new_size

[in] İlk boyut için belirlenen yeni büyüklük.

reserve_size=0

[in] Olası ilaveler için ayrılan büyüklük.

Dönüş değeri

Başarılı çalıştırma durumunda büyüklüğü ayarlanmış dizinin eleman sayısına dönüş yapar. Aksi durumda -1 değerine dönüş yapar ve dizi büyüklüğü değiştirilmez.

ArrayResize() fonksiyonu bir statik diziye, bir zaman serisine veya bir gösterge tamponuna uygulandığında büyüklük değerleri değişmez (bu tip diziler için yeni bellek tahsis edilmez). Bu durumda *yeni_boyut*<=ArraySize(*dizi*) işlemi *yeni_boyut* değerine; aksi durumda -1 değerine dönüş yapar.

Not

Bu fonksiyon sadece dinamik diziler üzerinde uygulanabilir. SetIndexBuffer() fonksiyonu aracılığıyla gösterge tamponu olarak atanmış dizilerin büyüklükleri değiştirilemez. Gösterge tamponları için tüm yeniden boyutlandırma işlemleri, terminalin çalışma zamanı alt sistemi tarafından gerçekleştirilir.

Dizideki elemanların sayısı 2147483647'yi geçemez.

Bellek tahsisi sık yapılıyorsa, fiziksel bellek tahsisinin sayısını azaltmak için, rezervi ayarlayan üçüncü parametrenin kullanımı tavsiye edilir. ArrayResize'ın daha sonraki çağrılarının tamamı, belleğin fiziksel olarak yeniden tahsisine yol açmaz. Sadece ilk boyutun büyüklüğünü ayrılmış bellek içerisinde değiştirir. Üçüncü parametrenin sadece fiziksel bellek tahsisi sırasında kullanılacağı hatırlanmalıdır. Örnek olarak:

```
ArrayResize(arr,1000,1000);
for(int i=1;i<3000;i++)
   ArrayResize(arr,i,1000);
```

Bu durumda belleğin yeniden tahsisi iki kere gerçekleşir. İlki, 3000 elemanlı döngüye girilmeden (dizi büyüklüğü 1000 olarak ayarlanacak); ikincisi ise i, 2000'e eşit olduğunda gerçekleşecektir. Üçüncü parametreyi atladığımız takdirde, belleğin 2000 defa yeniden tahsisi söz konusu olacaktır, bu da programı yavaşlatacaktır.

Örnek:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Sayaçlar
   ulong start=GetTickCount();
   ulong now;
   int   count=0;
//--- Hızlı bir versiyonun gösterimi için bir dizi oluştur
   double arr[];
   ArrayResize(arr,100000,100000);
//--- Bellek ayrılan versiyonun ne kadar hızlı çalıştığını kontrol et
   Print("--- Test Fast: ArrayResize(arr,100000,100000)");
   for(int i=1;i<=300000;i++)
     {
      //--- 100,000 elemanlı rezerve belirleyerek yeni dizi büyüklüğü ayarla!
      ArrayResize(arr,i,100000);
      //--- Yuvarlak bir sayıya ulaştığında , dizi büyüklüğünü ve harcanan zamanı göst
      if(ArraySize(arr)%100000==0)
        {
         now=GetTickCount();
         count++;
         PrintFormat("%d. ArraySize(arr)=%d Time=%d ms",count,ArraySize(arr),(now-sta
         start=now;
        }
     }
//--- Şimdi, rezerve bellek tahsisi olmayan versiyonun ne kadar yavaş çalıştığını göst
   double slow[];
   ArrayResize(slow,100000,100000);
//---
   count=0;
   start=GetTickCount();
   Print("---- Test Slow: ArrayResize(slow,100000)");
//---
   for(int i=1;i<=300000;i++)
     {
      //--- Ek rezerv olmadan yeni bir dizi büyüklüğü belirle
      ArrayResize(slow,i);
      //--- Yuvarlak bir sayıya ulaştığında , dizi büyüklüğünü ve harcanan zamanı göst
      if(ArraySize(slow)%100000==0)
        {
         now=GetTickCount();
         count++;
         PrintFormat("%d. ArraySize(slow)=%d Time=%d ms",count,ArraySize(slow),(now-st
         start=now;
        }
```

```
      }
   }
//--- Betiğin sonucu
/*
   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
*/
```

Ayrıca Bakınız

ArrayInitialize

ArrayInsert

Belirtilen eleman sayısını belirli bir indeksten başlayarak bir kaynak diziden bir alıcı diziye ekler.

```
bool  ArrayInsert(
   void&        dst_array[],          // alıcı dizi
   const void&  src_array[],          // kaynak dizi
   uint         dst_start,            // ekleme yapılacak alıcı dizi indeksi
   uint         src_start=0,          // kopyalama yapılacak kaynak dizi indeksi
   uint         count=WHOLE_ARRAY     // eklenecek eleman sayısı
   );
```

Parametreler

dst_array[]

[in][out] Elemanların ekleneceği alıcı dizi.

src_array[]

[in] Elemanların kopyalanacağı kaynak dizi.

dst_start

[in] Kaynak diziden eleman eklemek için alıcı dizideki indeks.

src_start=0

[in] Elemanların eklenmek üzere alındığı kaynak dizideki indeks.

count

[in] Kaynak diziden eklenecek eleman sayısı. WHOLE_ARRAY, belirtilen indeksten dizinin sonuna kadar olan tüm elemanları ifade eder.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata hakkında bilgi edinmek için, GetLastError() fonksiyonunu çağırın. Olası hatalar:

- 5052 - ERR_SMALL_ARRAY (*start* ve/veya *count* parametreleri yanlış olarak ayarlı veya *src_array[]* kaynak dizisi boş),
- 5056 - ERR_SERIES_ARRAY (dizi değiştirilemez, gösterge tamponu),
- 4006 - ERR_INVALID_ARRAY (kendine kopyalamaya izin verilmez veya diziler farklı tiptedir ya da sınıf nesneleri veya yıkıcı yapılar içeren sabit boyutlu bir dizi vardır.),
- 4005 - ERR_STRUCT_WITHOBJECTS_ORCLASS (Dizi POD yapısı içermez, bu durum basit kopyalamanın mümkün olmadığı anlamına gelir),
- *dst_array[]* alıcı dizi boyutunu değiştirirken meydana gelen hatalar, ArrayRemove() fonksiyon tanımında verilir.

Not

Fonksiyon sabit boyutlu bir dizi için kullanılıyorsa, *dst_array[]* alıcı dizinin kendisinin boyutu değişmez. *dst_start* konumundan başlayarak, alıcı dizinin elemanları sağa kaydırılır (elemanların son *count* parametresi "çıkar"), kaynak dizinden kopyalanan elemanlar ise onların yerlerini geçer.

Elemanları, SetIndexBuffer() fonksiyonu aracılığıyla gösterge tamponları olarak belirtilen dinamik dizilere ekleyemezsiniz. Gösterge tamponları için, tüm boyut değiştirme operasyonları terminalin yürütücü alt sistemi tarafından gerçekleştirilir.

Kaynak dizide, elemanlar *src_start* indeksinden başlayarak kopyalanır. Kaynak dizi boyutu değişmeden kalır. Alıcı diziye eklenecek elemanlar, kaynak dizi elemanlarına bağlantılar değildir. Bunun anlamı, iki diziden herhangi birindeki elemanların sonraki değişikliklerinin ikinci dizide yansıtılmamasıdır.

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- sabit boyutlu diziyi bildir ve değerleri doldur
   int array_dest[10];
   for(int i=0;i<10;i++)
     {
      array_dest[i]=i;
     }
   //--- kaynak dizi
   int array_source[10];
   for(int i=0;i<10;i++)
     {
      array_source[i]=10+i;
     }
//--- elemanları eklemeden önce dizileri göster
   Print("ArrayInsert() fonksiyonunu çağırmadan önce");
   ArrayPrint(array_dest);
   ArrayPrint(array_source);
//--- kaynak diziden 3 eleman ekle ve alıcı dizinin yeni kümesini göster
   ArrayInsert(array_dest,array_source,4,0,3);
   Print("ArrayInsert() fonksiyonunu çağırdıktan sonra");
   ArrayPrint(array_dest);
/*
  Gerçekleşim sonucu
   ArrayInsert() fonksiyonunu çağırmadan önce
   0 1 2 3 4 5 6 7 8 9
   ArrayInsert() fonksiyonunu çağırdıktan sonra
   0 1 2 3 10 11 12 7 8 9
*/
```

Ayrıca bakınız

ArrayRemove, ArrayCopy, ArrayResize, ArrayFree

ArrayRemove

Belirtilen eleman sayısını, belirtilen indeksten başlayarak diziden kaldırır.

```
bool  ArrayRemove(
   void&         array[],            // herhangi bir tip dizisi
   uint          start,              // kaldırmanın başladığı indeks
   uint          count=WHOLE_ARRAY   // eleman sayısı
   );
```

Parametreler

array[]

[in][out] Dizi.

start

[in] Dizi elemanlarının kaldırılmaya başlandığı indeks.

count=WHOLE_ARRAY

[in] Kaldırılan eleman sayısı. WHOLE_ARRAY değeri, tüm elemanların belirtilen indeksten dizinin sonuna kadar kaldırılması anlamına gelir.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata hakkında bilgi edinmek için, GetLastError() fonksiyonunu çağırın. Olası hatalar:

- 5052 - ERR_SMALL_ARRAY (aşırı büyük *start* değeri),
- 5056 - ERR_SERIES_ARRAY (dizi değiştirilemez, gösterge tamponu),
- 4003 - ERR_INVALID_PARAMETER (aşırı büyük *count* değeri),
- 4005 - ERR_STRUCT_WITHOBJECTS_ORCLASS (yıkıcıya sahip karmaşık nesneler içeren sabit boyutlu dizi),
- 4006 - ERR_INVALID_ARRAY (Bir yıkıcıya sahip yapı veya sınıf nesneleri içeren sabit boyutlu dizi).

Not

Fonksiyon sabit boyutlu bir dizi için kullanılıyorsa, dizi boyutu değişmez: kalan "kuyruk" fiziksel olarak *start* konumuna kopyalanır. Fonksiyonun nasıl çalıştığını doğru bir şekilde anlamak için aşağıdaki örneğe bakın. "Fiziksel" kopyalama, kopyalanan nesnelerin yapıcı veya kopyalama operatörü çağırarak oluşturulmadığı anlamına gelir. Bunun yerine, bir nesnenin ikili gösterimi kopyalanır. Bu nedenle, ArrayRemove() fonksiyonunu, yıkıcıya sahip nesneleri içeren sabit boyutlu diziye uygulayamazsınız (ERR_INVALID_ARRAY veya ERR_STRUCT_WITHOBJECTS_ORCLASS hatası etkinleştirilir). Böyle bir nesneyi kaldırırken, yıkıcı iki kere çağrılmalıdır - orijinal nesne ve kopyası için.

Elemanları, SetIndexBuffer() fonksiyonu aracılığıyla gösterge tamponları olarak belirtilen dinamik dizilerden kaldıramazsınız. Bu ERR_SERIES_ARRAY hatasına neden olur. Gösterge tamponları için, tüm boyut değiştirme operasyonları terminalin yürütücü alt sistemi tarafından gerçekleştirilir.

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
```

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- sabit boyutlu diziyi bildir ve değerleri doldur
   int array[10];
   for(int i=0;i<10;i++)
     {
      array[i]=i;
     }
//--- elemanları kaldırmadan önce diziyi göster
   Print("ArrayRemove() fonksiyonunu çağırmadan önce");
   ArrayPrint(array);
//--- dizideki 2 elemanı sil ve yeni grubu görüntüle
   ArrayRemove(array,4,2);
   Print("ArrayRemove() fonksiyonunu çağırdıktan sonra");
   ArrayPrint(array);
/*
  Gerçekleşim sonucu:
  ArrayRemove() fonksiyonunu çağırmadan önce
   0 1 2 3 4 5 6 7 8 9
  ArrayRemove() fonksiyonunu çağırdıktan sonra
   0 1 2 3 6 7 8 9 8 9
*/
```

Ayrıca bakınız

ArrayInsert, ArrayCopy, ArrayResize, ArrayFree

ArrayReverse

Dizideki belirtilen eleman sayısını, belirtilen indeksten başlayarak tersine çevirir.

```
bool  ArrayReverse(
   void&         array[],            // herhangi bir tip dizisi
   uint          start=0,            // dizinin tersine çevrilmeye başlanacağı indeks
   uint          count=WHOLE_ARRAY   // eleman sayısı
   );
```

Parametreler

array[]

[in][out] Dizi.

start=0

[in] Dizinin tersine çevriminin başladığı indeks.

count=WHOLE_ARRAY

[in] Ters çevrilmiş elemanların sayısı. Eğer WHOLE_ARRAY ise, tüm dizi elemanları belirtilen *start* indeksinden başlayarak dizinin sonuna kadar ters çevrilir.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner.

Not

ArraySetAsSeries() fonksiyonu, dizi elemanlarını fiziksel olarak hareket ettirmez. Bunun yerine, elemanlara erişimi zaman serilerinde olduğu gibi düzenlemek için yalnızca indeksleme yönünü geriye doğru değiştirir. ArrayReverse() fonksiyonu, dizi elemanlarını fiziksel olarak hareket ettirir, böylece dizi "tersine çevrilir".

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- sabit boyutlu diziyi bildir ve değerleri doldur
   int array[10];
   for(int i=0;i<10;i++)
     {
      array[i]=i;
     }
//--- elemanları tersine çevirmeden önce diziyi göster
   Print("ArrayReverse() fonksiyonunu çağırmadan önce");
   ArrayPrint(array);
//--- dizideki 3 elemanı tersine çevir ve yeni grubu göster
   ArrayReverse(array,4,3);
   Print("ArrayReverse() fonksiyonunu çağırdıktan sonra");
```

```
   ArrayPrint(array);
/*
  Gerçekleşim sonucu:
  ArrayReverse() fonksiyonunu çağırmadan önce
   0 1 2 3 4 5 6 7 8 9
  ArrayReverse() fonksiyonunu çağırdıktan sonra
   0 1 2 3 6 5 4 7 8 9
*/
```

Ayrıca bakınız

ArrayInsert, ArrayRemove, ArrayCopy, ArrayResize, ArrayFree, ArrayGetAsSeries, ArraySetAsSeries

ArraySetAsSeries

Belirtilen dinamik dizi nesnesi için AS_SERIES bayrağını ayarlar. Bu şekilde dizi elemanları zaman serisi şeklinde indislenir.

```
bool  ArraySetAsSeries(
   const void&  array[],     // referans dizi
   bool         flag         // 'true' değeri indisleme sırasının tersine çevrildiğini
   );
```

Parametreler

array[]

[in][out] Ayarlanacak sayısal dizi.

flag

[in] Dizi indisleme yönü.

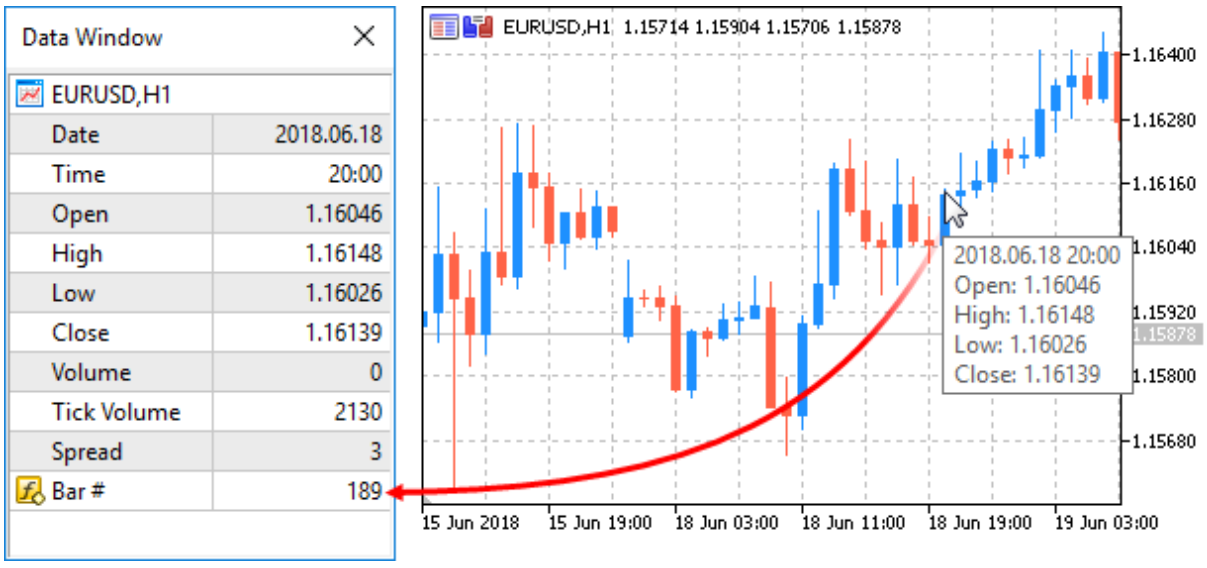
Dönüş değeri

Başarılı sonuç için 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Not

AS_SERIES bayrağı çok boyutlu diziler ve statik diziler (köşeli parantez içindeki büyükleri derleme aşamasında zaten ayarlanmış olan diziler) için ayarlanamaz. Zaman serilerindeki indisleme yönü sıradan dizilerden farklıdır. Sondan başa doğru (en yeni veriden en eski veriye doğru) indisleme yapılır.

Örnek: Çubuk numarasını gösteren gösterge




```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//---- Çizimlerin numaralandırılması
```

```
#property indicator_label1  "Numeration"
#property indicator_type1   DRAW_LINE
#property indicator_color1  CLR_NONE
//--- gösterge tamponları
double         NumerationBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,NumerationBuffer,INDICATOR_DATA);
//--- tamponların indis yönünü zaman serilerindeki gibi ayarla
   ArraySetAsSeries(NumerationBuffer,true);
//--- veri Penceresi içindeki gösterimin doğruluğunu ayarlayın
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//--- gösterge dizisinin ismi, Veri Penceresi içinde nasıl gözükecek
   PlotIndexSetString(0,PLOT_LABEL,"Bar #");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- mevcut sıfır çubuğunun açılış zamanını kaydet
   static datetime currentBarTimeOpen=0;
//--- time[] dizisinin erişim yönünü ters çevir - zaman serilerindeki gibi indisle
   ArraySetAsSeries(time,true);
//--- sıfır çubuğunun açılış zamanı kaydedilen zamandan farklıysa
   if(currentBarTimeOpen!=time[0])
     {
     //--- şimdikinden başlayarak çizelge derinliğine kadar tüm çubukları say
      for(int i=rates_total-1;i>=0;i--) NumerationBuffer[i]=i;
      currentBarTimeOpen=time[0];
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Ayrıca Bakınız

Zaman serilerine erişim, ArrayGetAsSeries

ArraySize

Belirtilen dizinin eleman sayısını verir.

```
int  ArraySize(
   const void&  array[]    // denetlenen dizi
   );
```

Parametreler

array[]

[in] Herhangi bir tipteki dizi.

Dönüş değeri

int tipli değer.

Not

Tek boyutlu bir dizi için ArraySize dönüş değeri ArrayRange(array,0) dönüş değerine eşit olacaktır.

Örnek:

```
void OnStart()
  {
//--- dizileri oluştur
   double one_dim[];
   double four_dim[][10][5][2];
//--- büyüklükler
   int one_dim_size=25;
   int reserve=20;
   int four_dim_size=5;
//--- yardımcı değişken
   int size;
//--- belleği yedekleme olmadan tahsis et
   ArrayResize(one_dim,one_dim_size);
   ArrayResize(four_dim,four_dim_size);
//--- 1. tek boyutlu dizi
   Print("+============================================================+");
   Print("Dizi büyüklükleri:");
   Print("1. Tek boyutlu dizi");
   size=ArraySize(one_dim);
   PrintFormat("Sıfır boyutunun büyüklüğü = %d, Dizi Büyüklüğü = %d",one_dim_size,size
//--- 2. çok boyutlu dizi
   Print("2. çok boyutlu dizi");
   size=ArraySize(four_dim);
   PrintFormat("Sıfır boyutunun büyüklüğü = %d, Dizi büyüklüğü = %d",four_dim_size,siz
//--- boyut büyüklükleri
   int d_1=ArrayRange(four_dim,1);
   int d_2=ArrayRange(four_dim,2);
   int d_3=ArrayRange(four_dim,3);
   Print("Kontrol et:");
```

```
   Print("Sıfır boyutu = Dizi büyüklüğü / (İlk boyut * İkinci boyut * Üçüncü boyut)");
   PrintFormat("%d = %d / (%d * %d * %d)",size/(d_1*d_2*d_3),size,d_1,d_2,d_3);
//--- 3. bellek yedeklemesiyle tek boyutlu dizi
   Print("3. Bellek yedeklemesiyle tek boyutlu dizi");
//--- değeri ikiye katla
   one_dim_size*=2;
//--- belleği yedekleme ile tahsis et
   ArrayResize(one_dim,one_dim_size,reserve);
//--- büyüklük değerini sonuca yaz
   size=ArraySize(one_dim);
   PrintFormat("Yedeklemeli büyüklük = %d, Dizinin gerçek büyüklüğü = %d",one_dim_size
  }
```

ArraySort

Çok boyutlu bir sayısal dizinin ilk boyutundaki elemanları artan şekilde sıralar.

```
bool  ArraySort(
   void&  array[]      // sıralanacak dizi
   );
```

Parametreler

array[]

[in][out] Sıralanacak sayısal dizi.

Dönüş değeri

Başarılı sonuç için 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Not

Dizi, AS_SERIES bayrağının değerine bakılmaksızın her zaman artan şekilde sıralanır.

ArraySort ve ArrayBSearch fonksiyonları tüm dizileri parametre olarak kabul eder. Ama arama ve sıralama işlemleri sadece ilk boyutta gerçekleştirilir.

Örnek:

```
#property description "Gösterge önceki ayın verisini inceleyerek, küçük ve büyük tik h
#property description "mumları boyar. Tik hacmi dizisi böyle mumları tanımlamak için s
#property description "Dizi elemanlarının ilk InpSmallVolume yüzdesini kapsayan hacme
#property description "mumlar küçük farz edilir. Dizi elemanlarının son InpBigVolume y
#property description "kapsayan hacme sahip mumlar ise büyük farz edilir."
//--- gösterge ayarları
#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   1
//--- grafik
#property indicator_label1  "VolumeFactor"
#property indicator_type1   DRAW_COLOR_CANDLES
#property indicator_color1  clrDodgerBlue,clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  2
//--- ön tanımlı sabitler
#define INDICATOR_EMPTY_VALUE 0.0
//--- giriş parametreleri
input int InpSmallVolume=15; // Küçük hacimlerin yüzdelik değeri (<50)
input int InpBigVolume=20;   // Büyük hacimlerin yüzdelik değeri (<50)
//--- analiz başlangıç zamanı (kaydırılacak)
datetime ExtStartTime;
//--- gösterge tamponları
double   ExtOpenBuff[];
double   ExtHighBuff[];
double   ExtLowBuff[];
```

```
double   ExtCloseBuff[];
double   ExtColorBuff[];
//--- mumların gösterilmesi için sınır hacim değerleri
long     ExtLeftBorder=0;
long     ExtRightBorder=0;
//+------------------------------------------------------------------+
//| Tik hacimleri için sınır değerlerini al                          |
//+------------------------------------------------------------------+
bool GetVolumeBorders(void)
  {
//--- değişkenler
   datetime stop_time;  // kopyalama durdurma zamanı
   long     buff[];     // kopyalama tamponu
//--- durdurma zamanı şu anki zaman olsun
   stop_time=TimeCurrent();
//--- başlangıç zamanı şimdikinden bir ay önceki olsun
   ExtStartTime=GetStartTime(stop_time);
//--- tik hacmi değerlerini al
   ResetLastError();
   if(CopyTickVolume(Symbol(),Period(),ExtStartTime,stop_time,buff)==-1)
     {
      //--- verinin alınması başarısız oldu, yeniden hesaplama komutu için false dönüş
      PrintFormat("Tik hacmi değerlerinin alınması başarısız oldu. Hata kodu = %d",Get
      return(false);
     }
//--- dizi büyüklüğünü hesapla
   int size=ArraySize(buff);
//--- diziyi sırala
   ArraySort(buff);
//--- tik hacimleri için sağ ve sol sınırları tanımla
   ExtLeftBorder=buff[size*InpSmallVolume/100];
   ExtRightBorder=buff[(size-1)*(100-InpBigVolume)/100];
//--- başarılı çalıştırma
   return(true);
  }
//+------------------------------------------------------------------+
//| Bir ay önceki verinin alınması                                   |
//+------------------------------------------------------------------+
datetime GetStartTime(const datetime stop_time)
  {
//--- durdurma zamanını MqlDateTime tipli yapı değişkenine dönüştür
   MqlDateTime temp;
   TimeToStruct(stop_time,temp);
//--- bir ay önceki veriyi al
   if(temp.mon>1)
      temp.mon-=1;  // mevcut ay, yılın ilk ayı değil. Bu yüzden, bir öncekinin numara
   else
     {
      temp.mon=12;  // mevcut ay, yılın ilk ayı. Bu yüzden, bir öncekinin numarası 12
```

```
      temp.year-=1; // yıl numarası bir küçükse
     }
//--- gün numarası 28'i aşamaz
   if(temp.day>28)
      temp.day=28;
//--- elde edilen tarih değerine dönüş yap
   return(StructToTime(temp));
  }
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- giriş parametreleri koşulları sağlıyor mu kontrol et
   if(InpSmallVolume<0 || InpSmallVolume>=50 || InpBigVolume<0 || InpBigVolume>=50)
     {
      Print("Hatalı giriş parametreleri");
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,ExtOpenBuff);
   SetIndexBuffer(1,ExtHighBuff);
   SetIndexBuffer(2,ExtLowBuff);
   SetIndexBuffer(3,ExtCloseBuff);
   SetIndexBuffer(4,ExtColorBuff,INDICATOR_COLOR_INDEX);
//--- gösterilmeyecek olan değeri ayarla
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALUE);
//--- gösterge tamponlarının etiketlerini ayarla
   PlotIndexSetString(0,PLOT_LABEL,"Open;High;Low;Close");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- hala işlenmeyen çubuklar var mı kontrol et
   if(prev_calculated<rates_total)
     {
```

```
      //--- sağ ve sol hacim sınırları için yeni değerler al
      if(!GetVolumeBorders())
         return(0);
     }
//--- çubuk hesaplama için değişkeni başlat
   int start=prev_calculated;
//--- gösterge değerleri bir önceki tik ile zaten hesaplanmışsa son çubukta çalış
   if(start>0)
      start--;
//--- zaman serilerindeki doğrudan indislemeyi ayarla
   ArraySetAsSeries(time,false);
   ArraySetAsSeries(open,false);
   ArraySetAsSeries(high,false);
   ArraySetAsSeries(low,false);
   ArraySetAsSeries(close,false);
   ArraySetAsSeries(tick_volume,false);
//--- gösterge değerlerini hesaplayan döngü
   for(int i=start;i<rates_total;i++)
     {
      //--- başlangıç tarihinden başlayarak mumları doldur
      if(ExtStartTime<=time[i])
        {
         //--- değer sağ sınırdan küçük değilse, mumu doldur
         if(tick_volume[i]>=ExtRightBorder)
           {
            //--- mumu çizmek için veri al
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- DodgerBlue (parlak gök mavisi) renk
            ExtColorBuff[i]=0;
            //--- döngüye devam et
            continue;
           }
         //--- değer, sol sınırı aşmamışsa, mumu doldur
         if(tick_volume[i]<=ExtLeftBorder)
           {
            //--- mumu çizmek için veri al
            ExtOpenBuff[i]=open[i];
            ExtHighBuff[i]=high[i];
            ExtLowBuff[i]=low[i];
            ExtCloseBuff[i]=close[i];
            //--- Orange (turuncu) renk
            ExtColorBuff[i]=1;
            //--- döngüye devam et
            continue;
           }
        }
```

```
      //--- hesaplamaya katılmayan boş değerli çubukları ayarla
      ExtOpenBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtHighBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtLowBuff[i]=INDICATOR_EMPTY_VALUE;
      ExtCloseBuff[i]=INDICATOR_EMPTY_VALUE;
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Ayrıca Bakınız

ArrayBsearch

ArraySwap

Aynı türdeki iki dinamik dizinin içeriğini değiştirir. Çok boyutlu diziler için, ilk boyut haricindeki tüm boyutlardaki öğe sayısı eşleşmelidir.

```
bool  ArraySwap(
   void&  array1[],      // ilk dizi
   void&  array2[]       // ikinci dizi
   );
```

Parameters

array1[]

[in][out] Sayısal türün dizisi.

array2[]

[in][out] Sayısal türün dizisi.

Dönen değer

Eğer başarılı ise true, aksi halde false döndürür. Bu durumda, GetLastError(),ERR_INVALID_ARRAY hata kodunu döndürür.

Not

Fonksiyon, ilki hariç aynı türdeki ve aynı boyuttaki dinamik dizileri kabul eder. Tam sayı türüleri için, işaret gözardı edilir, örneğin char==uchar)

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- tekliflerin depolanması için diziler
   double source_array[][8];
   double   dest_array[][8];
   MqlRates rates[];
//--- Geçerli zaman aralığında son 20 mumun verilerini alınması
   int copied=CopyRates(NULL,0,0,20,rates);
   if(copied<=0)
     {
      PrintFormat("CopyRates(%s,0,0,20,rates) failed, error=%d",
                  Symbol(),GetLastError());
      return;
     }
//--- Kopyalanan veri miktarı için dizi boyutunu ayarlanması
   ArrayResize(source_array,copied);
//--- rate_array_1[] dizisini, rates[] den verileri kullanarak doldurulması
   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;
     }
//--- source_array[] ve dest_array[] arasındaki verileri değiştirilmesi
   if(!ArraySwap(source_array,dest_array))
     {
      PrintFormat("ArraySwap(source_array,rate_array_2) failed, error code=%d",GetLast
      return;
     }
//--- değiştirme sonrasında kaynak dizinin sıfır olmasının sağlanması
   PrintFormat("ArraySwap() done: ArraySize(source_array)=%d",ArraySize(source_array))
//--- dest_array[] hedef dizisinin verilerinin görüntülenmesi
   ArrayPrint(dest_array);
  }
```

Ayrıca bakın

ArrayCopy, ArrayFill, ArrayRange, ArrayIsDynamic

Matrisler ve Vektörler

Bir matris, double, float veya complex türünde sayılardan oluşan iki boyutlu bir dizidir.

Bir vektör, double, float veya complex türünde sayılardan oluşan tek boyutlu bir dizidir. Vektörlerin dikey mi yoksa yatay mı olduğuna dair bir işareti yoktur. Kullanım durumuna göre belirlenir. Örneğin, Dot vektör işlemi, soldaki vektörün yatay, sağdaki vektörün ise dikey olduğunu varsayar. Belirli bir yön gerekiyorsa, tek satırlı veya tek sütunlu matrisler kullanılabilir. Ancak genellikle bu gerekli değildir.

Matrisler ve vektörler, veriler için dinamik olarak bellek tahsis eder. Aslında matrisler ve vektörler, büyüklük ve içerdikleri veri türü gibi belirli özelliklere sahip nesnelerdir. Matrislerin ve vektörlerin özellikleri, vector_a.Size(), matrix_b.Rows(), vector_c.Norm(), matrix_d.Cond() vb. metotlar kullanılarak elde edilebilir. Ayrıca herhangi bir büyüklük de değiştirilebilir.

Matrisler oluşturulurken ve başlatılırken, statik metotlar kullanılır (bunlar bir sınıfın statik metotları gibidir). Örneğin: matrix::Eye(), matrix::Identity(), matrix::Ones(), vector::Ones(), matrix::Zeros(), vector::Zeros(), matrix::Full(), vector::Full(), matrix::Tri().

Şu anda, matrisler ve vektörler üzerindeki işlemler, complex veri türünün kullanımını içermemektedir - bu veri türü üzerindeki çalışmalar henüz tamamlanmamıştır.

MQL5, matrislerin ve vektörlerin DLL'lere aktarılmasını destekler. Bu, ilgili veri türlerini kullanan fonksiyonların harici kütüphanelerden içe aktarılmasına olanak sağlar.

Matrisler ve vektörler DLL'e arabellek işaretçisi olarak aktarılır. Örneğin, float türünde bir matrisin aktarılması için, DLL'den dışa aktarılan fonksiyonun ilgili parametresinin float türünde arabellek işaretçisi alması gerekir.

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
```

Matrislerin ve vektörlerin düzgün bir şekilde işlenmesi adına, arabelleklere ek olarak büyüklüklerinin de aktarılması gerekir.

Tüm matris ve vektör metotları aşağıda alfabetik sırayla listelenmiştir.

Fonksiyon	Eylem	Kategori
Activation	Aktivasyon fonksiyonu değerlerini hesaplar ve onları aktarılan vektöre/matrise yazar	Makine öğrenimi
ArgMax	Maksimum değerin indeksini geri döndürür	İstatistikler
ArgMin	Minimum değerin indeksini geri döndürür	İstatistikler

Fonksiyon	Eylem	Kategori
ArgSort	Sıralanmış indeksi geri döndürür	Manipülasyonlar
Assign	Otomatik dönüştürmeyle matrisi, vektörü veya diziyi kopyalar	Başlatma
Average	Matris/vektör değerlerinin ağırlıklı ortalamasını hesaplar	İstatistikler
Cholesky	Cholesky ayrışmasını hesaplar	Dönüşümler
Clip	Matrisin/vektörün elemanlarını belirli bir geçerli değerler aralığıyla sınırlar	Manipülasyonlar
Col	Bir sütun vektörü geri döndürür. Belirtilen sütuna vektörü yazar	Manipülasyonlar
Cols	Matristeki sütun sayısını geri döndürür	Özellikler
Compare	Belirtilen hassasiyetle iki matrisin/vektörün elemanlarını karşılaştırır	Manipülasyonlar
CompareByDigits	Anlamlı basamak hassasiyetiyle iki matrisin/vektörün elemanlarını karşılaştırır	Manipülasyonlar
Cond	Matrisin koşul sayısını hesaplar	Özellikler
Convolve	İki vektörün ayrık, lineer konvolüsyonunu geri döndürür	Çarpımlar
Copy	Matrisin/vektörün bir kopyasını geri döndürür	Manipülasyonlar
CopyIndicatorBuffer	Belirtilen gösterge arabelleğinden belirtilen miktarda veriyi vektöre alır	Başlatma
CopyRates	Belirtilen sembol, zaman dilimi ve veri miktarı için MqlRates yapısından matrise veya vektöre geçmiş seriler alır	Başlatma
CopyTicks	MqlTick yapısından matrise veya vektöre tikler alır	Başlatma
CopyTicksRange	Belirtilen tarih aralığı için MqlTick yapısından matrise veya vektöre tikler alır	Başlatma
CorrCoef	Pearson korelasyon katsayısını (lineer korelasyon katsayısı)	Çarpımlar

Fonksiyon	Eylem	Kategori
	hesaplar	
Correlate	İki vektörün çapraz korelasyonunu hesaplar	Çarpımlar
Cov	Kovaryans matrisini hesaplar	Çarpımlar
CumProd	Belirtilen eksen boyunca olanlar da dahil olmak üzere matris/vektör elemanlarının kümülatif çarpımını geri döndürür	İstatistikler
CumSum	Belirtilen eksen boyunca olanlar da dahil olmak üzere matris/vektör elemanlarının kümülatif toplamını geri döndürür	İstatistikler
Derivative	Aktivasyon fonksiyonu türevinin değerlerini hesaplar ve onları aktarılan vektöre/matrise yazar	Makine öğrenimi
Det	Tersinir kare matrisin determinantını hesaplar	Özellikler
Diag	Bir köşegeni çıkarır veya bir köşegen matris oluşturur	Manipülasyonlar
Dot	İki vektörün nokta çarpımı	Çarpımlar
Eig	Kare matrisin özdeğerlerini ve sağ özvektörlerini hesaplar	Dönüşümler
EigVals	Genel matrisin özdeğerlerini hesaplar	Dönüşümler
Eye	Köşegende 1'lerin ve diğer yerlerde 0'ların olduğu bir matris geri döndürür	Başlatma
Fill	Matrisi veya vektörü belirtilen değerle doldurur	Başlatma
Flat	Matris elemanının iki yerine bir indeks aracılığıyla adreslenmesine olanak sağlar	Manipülasyonlar
Full	Belirtilen değerle dolu yeni bir matris oluşturur ve geri döndürür	Başlatma
GeMM	Genel matris çarpımı (General Matrix Multiply, GeMM)	Çarpımlar

Fonksiyon	Eylem	Kategori
	yöntemi, iki matrisin genel çarpımını uygular	
HasNan	Matris/vektördeki NaN değerlerinin sayısını geri döndürür	Manipülasyonlar
Hsplit	Bir matrisi yatay olarak birden çok alt matrise böler. axis=0 Split ile aynıdır	Manipülasyonlar
Identity	Belirtilen büyüklükte bir birim matris oluşturur	Başlatma
Init	Matrisi veya vektörü başlatır	Başlatma
Inner	İki matrisin iç çarpımı	Çarpımlar
Inv	Jordan-Gauss yöntemiyle tersinir kare matrisin çarpımsal tersini hesaplar	Çözümler
Kron	İki matrisin, matrisin ve vektörün, vektörün ve matrisin veya iki vektörün Kronecker çarpımını geri döndürür	Çarpımlar
LinearRegression	Hesaplanan lineer regresyon değerleriyle bir vektör/matris hesaplar	İstatistikler
Loss	Kayıp fonksiyonu değerlerini hesaplar ve onları aktarılan vektöre/matrise yazar	Makine öğrenimi
LstSq	Lineer cebirsel denklem sisteminin en küçük kareler çözümünü geri döndürür (kare olmayan veya dejenere matrisler için)	Çözümler
LU	Alt üçgen matris ve üst üçgen matrisin çarpımı olarak, matrisin LU ayrışması	Dönüşümler
LUP	Yalnızca satır permütasyonlarıyla LU ayrışması olarak, kısmi pivotlu LUP ayrışması: PA=LU	Dönüşümler
MatMul	İki matrisin matris çarpımı	Çarpımlar
Max	Matristeki/vektördeki maksimum değeri geri döndürür	İstatistikler

Fonksiyon	Eylem	Kategori
Mean	Elemanların aritmetik ortalamasını hesaplar	İstatistikler
Median	Matris/vektör elemanlarının medyanını hesaplar	İstatistikler
Min	Matristeki/vektördeki minimum değeri geri döndürür	İstatistikler
Norm	Matrisin veya vektörün normunu geri döndürür	Özellikler
Ones	1'lerle dolu yeni bir matris oluşturur ve geri döndürür	Başlatma
Outer	İki matrisin veya iki vektörün dış çarpımını hesaplar	Çarpımlar
Percentile	Matris/vektör elemanlarının veya belirtilen eksen boyuncaki elemanların belirtilen yüzdelik dilimini geri döndürür	İstatistikler
PInv	Moore-Penrose yöntemiyle matrisin yalancı tersini hesaplar	Çözümler
Power	Kare matrisi bir tam sayı kuvvete yükseltir	Çarpımlar
Prod	Belirtilen eksen boyunca da gerçekleştirilebilen matris/vektör elemanlarının çarpımını geri döndürür	İstatistikler
Ptp	Matrisin/vektörün veya belirtilen matris ekseninin değer aralığını geri döndürür	İstatistikler
QR	Matrisin qr ayrışmasını hesaplar	Dönüşümler
Quantile	Matris/vektör elemanlarının veya belirtilen eksen boyuncaki elemanların belirtilen dağılım dilimini geri döndürür	İstatistikler
Rank	Gauss yöntemini kullanarak matris rankını geri döndürür	Özellikler
RegressionMetric	Belirtilen veri kümesi üzerinde oluşturulan regresyon çizgisinden sapma hatası olarak, regresyon metriğini hesaplar	İstatistikler

Fonksiyon	Eylem	Kategori
Reshape	Verilerini değiştirmeden matrisin şeklini değiştirir	Manipülasyonlar
Resize	Belirtilen şekle ve büyüklüğe sahip yeni bir matris geri döndürür	Manipülasyonlar
Row	Bir satır vektörü geri döndürür. Belirtilen satıra vektörü yazar	Manipülasyonlar
Rows	Matristeki satır sayısını geri döndürür	Özellikler
Set	Belirtilen indekse göre vektör elemanının değerini ayarlar	Manipülasyonlar
Size	Vektörün büyüklüğünü geri döndürür	Özellikler
SLogDet	Matrisin determinantının işaretini ve logaritmasını hesaplar	Özellikler
Solve	Lineer matris denklemini veya lineer cebirsel denklem sistemini çözer	Çözümler
Sort	Matris veya vektörü yerinde sıralar	Manipülasyonlar
Spectrum	AT*A çarpımının özdeğerlerinin kümesi olarak, matrisin spektrumunu hesaplar	Özellikler
Split	Bir matrisi birden çok alt matrise böler	Manipülasyonlar
Std	Matris/vektör elemanlarının veya belirtilen eksen boyuncaki elemanların standart sapmasını geri döndürür	İstatistikler
Sum	Belirtilen eksen(ler) boyunca da gerçekleştirilebilen matris/vektör elemanlarının toplamını geri döndürür	İstatistikler
SVD	Tekil değer ayrışması	Dönüşümler
SwapCols	Matristeki sütunları değiştirir	Manipülasyonlar
SwapRows	Matristeki satırları değiştirir	Manipülasyonlar
Trace	Matrisin köşegenlerinin toplamını geri döndürür	Özellikler

Fonksiyon	Eylem	Kategori
Transpose	Matrisin devriğini alır (eksenleri değiştirir) ve değiştirilmiş matrisi geri döndürür	Manipülasyonlar
Tri	İstenilen köşegende ve altında 1'lerin ve diğer yerlerde 0'ların olduğu bir matris oluşturur	Başlatma
TriL	k'ıncı köşegenin üzerindeki elemanların 0'a çevrildiği matrisin bir kopyasını geri döndürür. Alt üçgen matris	Manipülasyonlar
TriU	k'ıncı köşegenin altındaki elemanların 0'a çevrildiği matrisin bir kopyasını geri döndürür. Üst üçgen matris	Manipülasyonlar
Var	Matris/vektör elemanlarının varyansını hesaplar	İstatistikler
Vsplit	Bir matrisi dikey olarak birden çok alt matrise böler. axis=1 Split ile aynıdır	Manipülasyonlar
Zeros	0'larla dolu yeni bir matris oluşturur ve geri döndürür	Başlatma

Matris ve Vektör Türleri

Matris ve vektör, MQL5'te lineer cebir işlemlerini mümkün kılan özel veri türleridir. Aşağıdaki veri türleri mevcuttur:

- matrix - double türü elemanlar içeren matris.
- matrixf - float türü elemanlar içeren matris.
- matrixc - complex türü elemanlar içeren matris.
- vector - double türü elemanlar içeren vektör.
- vectorf - float türü elemanlar içeren vektör.
- vectorc - complex türü elemanlar içeren vektör.

Şablon fonksiyonları, ilgili türler yerine matrix<double>, matrix<float>, vector<double>, vector<float> gibi gösterimleri desteklemektedir.

Matris ve vektör başlatma metotları

Fonksiyon	Eylem
Eye	Köşegende 1'lerin ve diğer yerlerde 0'ların olduğu bir matris geri döndürür
Identity	Belirtilen büyüklükte bir birim matris oluşturur
Ones	1'lerle dolu yeni bir matris oluşturur ve geri döndürür
Zeros	0'larla dolu yeni bir matris oluşturur ve geri döndürür
Full	Belirtilen değerle dolu yeni bir matris oluşturur ve geri döndürür
Tri	İstenilen köşegende ve altında 1'lerin ve diğer yerlerde 0'ların olduğu bir matris oluşturur
Init	Matrisi veya vektörü başlatır
Fill	Matrisi veya vektörü belirtilen değerle doldurur

Matris ve Vektör İşlemleri için Numaralandırmalar

Bu bölümde, çeşitli matris ve vektör metotlarında kullanılan numaralandırmalar açıklanmaktadır.

ENUM_AVERAGE_MODE

Yumuşatma türü numaralandırması.

Kimlik	Açıklama
AVERAGE_NONE	Ortalama alma yok. Sonuçlar her etiket için ayrı ayrı sağlanır
AVERAGE_BINARY	İkili sınıflandırma için etiket 1 sonucu
AVERAGE_MICRO	Ortalama hata matrisi sonucu (karışıklık matrisi)
AVERAGE_MACRO	Her bir etiketin hata matrislerinin sonuçlarından elde edilen ortalama sonuç
AVERAGE_WEIGHTED	Ağırlıklı ortalama sonuç

ENUM_VECTOR_NORM

vector::Norm için vektör normlarının numaralandırması.

Kimlik	Açıklama
VECTOR_NORM_INF	inf-normu
VECTOR_NORM_MINUS_INF	-inf-normu
VECTOR_NORM_P	p-normu

ENUM_MATRIX_NORM

matrix::Norm için ve matrix::Cond matris koşul sayısını elde etmek için matris normlarının numaralandırması.

Kimlik	Açıklama
MATRIX_NORM_FROBENIUS	Frobenius normu
MATRIX_NORM_SPECTRAL	Spektral norm
MATRIX_NORM_NUCLEAR	Nükleer norm
MATRIX_NORM_INF	inf-normu
MATRIX_NORM_P1	1-normu

Kimlik	Açıklama
MATRIX_NORM_P2	2-normu
MATRIX_NORM_MINUS_INF	-inf-normu
MATRIX_NORM_MINUS_P1	-1-normu
MATRIX_NORM_MINUS_P2	-2-normu

ENUM_VECTOR_CONVOLVE

Konvolüsyon vector::Convolve ve çapraz korelasyon vector::Correlate için numaralandırma.

Kimlik	Açıklama
VECTOR_CONVOLVE_FULL	full konvolüsyonu
VECTOR_CONVOLVE_SAME	same konvolüsyonu
VECTOR_CONVOLVE_VALID	valid konvolüsyonu

ENUM_REGRESSION_METRIC

vector::RegressionMetric için regresyon metriklerinin numaralandırması.

Kimlik	Açıklama
REGRESSION_MAE	Hataların mutlak değerlerinin ortalaması
REGRESSION_MSE	Hataların karelerinin ortalaması
REGRESSION_RMSE	Hataların karelerinin ortalamasının karekökü
REGRESSION_R2	R-kare
REGRESSION_MAPE	Yüzde olarak MAE
REGRESSION_MSPE	Yüzde olarak MSE
REGRESSION_RMSLE	Logaritmik ölçekte hesaplanan RMSE
REGRESSION_SMAPE	Simetrik MAPE
REGRESSION_MAXE	Hataların mutlak değerlerinin maksimumu
REGRESSION_MEDE	Hataların mutlak değerlerinin medyanı
REGRESSION_MPD	Ortalama Poisson sapması
REGRESSION_MGD	Ortalama gama sapması
REGRESSION_EXPV	Açıklanan varyans

ENUM_CLASSIFICATION_METRIC

Sınıflandırma problemleri için metriklerin numaralandırılması.

Kimlik	Açıklama
CLASSIFICATION_ACCURACY	Tüm sınıflar için tahmin doğruluğu açısından model kalitesi
CLASSIFICATION_AVERAGE_PRECISION	Ortalama model doğruluğu
CLASSIFICATION_BALANCED_ACCURACY	Dengeli tahmin doğruluğu
CLASSIFICATION_F1	F1 skoru. Model hassasiyeti ve geri çağırma arasındaki harmonik ortalama
CLASSIFICATION_JACCARD	Jaccard skoru
CLASSIFICATION_PRECISION	Hedef sınıf için gerçek pozitifleri tahmin etmede model doğruluğu
CLASSIFICATION_RECALL	Model tamlığı
CLASSIFICATION_ROC_AUC	Hata eğrisinin altındaki alan
CLASSIFICATION_TOP_K_ACCURACY	Doğru etiketin tahmin edilen k etiketin üstünde görünme sıklığı

ENUM_LOSS_FUNCTION

Kayıp fonksiyonu hesaplamaları vector::Loss için numaralandırma.

Kimlik	Açıklama
LOSS_MSE	Hataların karelerinin ortalaması
LOSS_MAE	Hataların mutlak değerlerinin ortalaması
LOSS_CCE	Kategorik çapraz entropi
LOSS_BCE	İkili çapraz entropi
LOSS_MAPE	Yüzde olarak MAE
LOSS_MSLE	Logaritmik ölçekte hesaplanan MSE
LOSS_KLD	Kullback-Leibler diverjansı
LOSS_COSINE	Kosinüs benzerliği/yakınlığı
LOSS_POISSON	Poisson
LOSS_HINGE	Hinge

Kimlik	Açıklama
LOSS_SQ_HINGE	Karesel hinge
LOSS_CAT_HINGE	Kategorik hinge
LOSS_LOG_COSH	Hiperbolik kosinüsün logaritması
LOSS_HUBER	Huber

ENUM_ACTIVATION_FUNCTION

Aktivasyon fonksiyonu vector::Activation ve aktivasyon fonksiyonu türev vector::Derivative için numaralandırma.

Kimlik	Açıklama
AF_ELU	Üssel lineer birim
AF_EXP	Üssel
AF_GELU	Gauss hata lineer birimi
AF_HARD_SIGMOID	Sert sigmoid
AF_LINEAR	Lineer
AF_LRELU	Sızıntılı düzeltilmiş lineer birim
AF_RELU	Düzeltilmiş lineer birim
AF_SELU	Ölçekli Üssel Lineer Birim
AF_SIGMOID	Sigmoid
AF_SOFTMAX	Softmax
AF_SOFTPLUS	Softplus
AF_SOFTSIGN	Softsign
AF_SWISH	Swish
AF_TANH	Hiperbolik tanjant fonksiyonu
AF_TRELU	Eşikli düzeltilmiş lineer birim

ENUM_SORT_MODE

Sort fonksiyonu için sıralama türlerinin numaralandırması.

Kimlik	Açıklama
SORT_MODE_ASCENDING	Artan düzende sırala

Kimlik	Açıklama
SORT_MODE_DESCENDING	Azalan düzende sırala

ENUM_MATRIX_AXIS

Matrisler için tüm istatistiksel fonksiyonlarda ekseni belirtmek için numaralandırma.

Kimlik	Açıklama
AXIS_NONE	Eksen belirtilmedi. Hesaplama, sanki bir vektörmüş gibi tüm matris elemanları üzerinden yapılır (Flat metoduna bakın)
AXIS_HORZ	Yatay eksen
AXIS_VERT	Dikey eksen

Başlatma

Matrisleri ve vektörleri bildirmenin ve başlatmanın birkaç yolu vardır.

Fonksiyon	Eylem
Assign	Otomatik dönüştürmeyle matrisi, vektörü veya diziyi kopyalar
CopyIndicatorBuffer	Belirtilen gösterge arabelleğinden belirtilen miktarda veriyi vektöre alır
CopyRates	Belirtilen sembol, zaman dilimi ve veri miktarı için MqlRates yapısından matrise veya vektöre geçmiş seriler alır
CopyTicks	MqlTick yapısından matrise veya vektöre tikler alır
CopyTicksRange	Belirtilen tarih aralığı için MqlTick yapısından matrise veya vektöre tikler alır
Eye	Köşegende 1'lerin ve diğer yerlerde 0'ların olduğu bir matris geri döndürür
Identity	Belirtilen büyüklükte bir birim matris oluşturur
Ones	1'lerle dolu yeni bir matris oluşturur ve geri döndürür
Zeros	0'larla dolu yeni bir matris oluşturur ve geri döndürür
Full	Belirtilen değerle dolu yeni bir matris oluşturur ve geri döndürür
Tri	İstenilen köşegende ve altında 1'lerin ve diğer yerlerde 0'ların olduğu bir matris oluşturur
Init	Matrisi veya vektörü başlatır
Fill	Matrisi veya vektörü belirtilen değerle doldurur

Büyüklük belirtilmeden bildirim (veriler için bellek tahsisi yoktur):

```
matrix          matrix_a;   // double türünde matris
matrix<double>  matrix_a1;  // double türünde matris bildirmenin başka bir yolu; şabl
matrixf         matrix_a2;  // float türünde matris
matrix<float>   matrix_a3;  // float türünde matris
vector          vector_a;   // double türünde vektör
vector<double>  vector_a1;
vectorf         vector_a2;  // float türünde vektör
vector<float>   vector_a3;
```

Belirtilen büyüklükle bildirim (veriler için bellek tahsisiyle, ancak herhangi bir başlatma olmadan):

```
matrix          matrix_a(128,128);          // parametreler sabit olabilir
matrix<double>  matrix_a1(InpRows,InpCols); // parametreler değişken olabilir
matrixf         matrix_a2(1,128);           // yatay vektörün analoğu
```

```
  matrix<float>  matrix_a3(InpRows,1);        // dikey vektörün analoğu
  vector         vector_a(256);
  vector<double> vector_a1(InpSize);
  vectorf        vector_a2(SomeFunc());       // SomeFunc fonksiyonu, vektör büyüklüğü
  vector<float>  vector_a3(InpSize+16);       // ifade parametre olarak kullanılabilir
```

Başlatmayla bildirim (matris ve vektör büyüklükleri başlatma sırasında belirlenir):

```
  matrix         matrix_a={{0.1,0.2,0.3},{0.4,0.5,0.6}};
  matrix<double> matrix_a1=matrix_a;                    // aynı türden matrisler olr
  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;                   // aynı türden vektörler olr
```

Başlatmayla bildirim:

```
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); // önce size_m x size_k büyü
  matrixf matrix_a1(10,20,MatrixArange);                  // float türünde bir matris
  vector  vector_a(size,VectorArange,-10.0);              // vektör oluşturulduktan sc
  vectorf vector_a1(128,VectorArange);
```

Veriler için bellek her zaman dinamik olduğundan, matris veya vektör büyüklüklerinin değiştirilebileceğini lütfen unutmayın.

Statik metotlar

Belirli bir şekilde başlatılan, belirtilen büyüklükte matrisler ve vektörler oluşturmak için statik metotlar:

```
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);
```

Önceden oluşturulmuş matrisleri ve vektörleri başlatma metotları:

```
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

Otomatik dönüştürmeyle matrisi, vektörü veya diziyi kopyalar.

```
bool matrix::Assign(
  const matrix<T>  &mat     // kopyalanacak matris
   );
bool matrix::Assign(
  const void       &array[] // kopyalanacak dizi
   );
bool vector::Assign(
  const vector<T>  &vec     // kopyalanacak vektör
   );
bool vector::Assign(
  const void       &array[] // kopyalanacak dizi
   );
```

Parametreler

m, v veya array

[in] Kopyalanacak matris, vektör veya dizi.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Not

Copy'den farklı olarak, Assign metodu dizilerin kopyalanmasına da olanak sağlar. Bu durumda, otomatik dönüştürme gerçekleşir, böylece ortaya çıkan matris veya vektör kopyalanan dizinin büyüklüğüne göre ayarlanır.

Örnek:

```
//--- matrisler kopyala
  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);

//--- matrise dizi kopyala
  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]]

*/
```

Ayrıca bakınız

Copy

CopyIndicatorBuffer

Belirtilen gösterge arabelleğinden belirtilen miktarda veriyi vektöre alır.

Veriler, vektör için tahsis edilen fiziksel belleğin başında en eski eleman olacak şekilde vektöre kopyalanacaktır. Fonksiyonun üç seçeneği vardır.

Başlangıç konumuna ve gerekli eleman sayısına göre erişim

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // gösterge tanıtıcısı
   ulong     buffer_index,         // gösterge arabelleği numarası
   ulong     start_pos,            // kopyalamanın başlayacağı konum
   ulong     count                 // kopyalanacak eleman sayısı
   );
```

Başlangıç tarihine ve gerekli eleman sayısına göre erişim

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // gösterge tanıtıcısı
   ulong     buffer_index,         // gösterge arabelleği numarası
   datetime  start_time,           // başlangıç tarihi
   ulong     count                 // kopyalanacak eleman sayısı
   );
```

Gerekli tarih aralığının başlangıç ve bitiş tarihlerine göre erişim

```
bool  vector::CopyIndicatorBuffer(
   long      indicator_handle,     // gösterge tanıtıcısı
   ulong     buffer_index,         // gösterge arabelleği numarası
   datetime  start_time,           // başlangıç tarihi
   datetime  stop_time             // bitiş tarihi
   );
```

Parametreler

indicator_handle

[in] İlgili gösterge fonksiyonu tarafından elde edilen gösterge tanıtıcısı.

buffer_index

[in] Gösterge arabelleği numarası.

start_pos

[in] Kopyalanacak ilk elemanın indeksi.

count

[in] Kopyalanacak eleman sayısı.

start_time

[in] İlk elemana karşılık gelen çubuk zamanı.

stop_time

[in] Son elemana karşılık gelen çubuk zamanı.

Geri dönüş değeri

Başarılı olursa true, hata olması durumunda false geri döndürür.

Not

Kopyalanacak veri elemanları (buffer_index indeksli gösterge arabelleği) şimdiki zamandan geçmişe doğru sayılır, yani 0'a eşit başlangıç konumu mevcut çubuk (mevcut çubuk için gösterge değeri) anlamına gelir.

Bilinmeyen miktarda veri kopyalarken, büyüklük belirtmeden (veriler için bellek tahsis etmeden) vektörü bildirmelisiniz, çünkü CopyBuffer() fonksiyonu alıcı vektörün büyüklüğünü kopyalanan veri büyüklüğüne göre ayarlayacaktır.

Gösterge değerlerinin kısmi olarak kopyalanması gerektiğinde, gerekli miktarın kopyalanacağı bir ara vektör kullanmalısınız. Daha sonra bu ara vektörden gerekli değerleri alıcı vektörün gerekli yerlerine eleman eleman kopyalayabilirsiniz.

Belirli bir miktarda veri kopyalanacaksa, gereksiz bellek yeniden tahsisini önlemek adına büyüklüğünü belirterek vektörü bildirmeniz önerilir.

Göstergeden veri talep edilirken, istenen seriler henüz oluşturulmadıysa veya sunucudan indirilmesi gerekiyorsa, fonksiyon hemen false geri döndürür ve arka planda indirme/oluşturma işlemini başlatır.

Uzman Danışmandan veya komut dosyasından veri talep edilirken, terminalde yerel olarak ilgili veriler yoksa, sunucudan indirme işlemi başlatılır ve bu süreçte yerel geçmişten oluşturulabiliyorsa gerekli seriler oluşturulur. Fonksiyon, zaman aşımı süresi dolduğunda hazır olan veri miktarını geri döndürür.

Ayrıca bakınız

CopyBuffer

CopyRates

Belirtilen sembol, zaman dilimi ve veri miktarı için MqlRates yapısından matrise veya vektöre geçmiş seriler alır. Elemanlar şimdiki zamandan geçmişe doğru sayılır, yani 0'a eşit başlangıç konumu mevcut çubuk anlamına gelir.

Veriler, en eski eleman matrisin/vektörün başına gelecek şekilde kopyalanır. Fonksiyonun üç seçeneği vardır.

Başlangıç konumuna ve gerekli eleman sayısına göre erişim

```
bool  CopyRates(
   string           symbol,       // sembol adı
   ENUM_TIMEFRAMES  period,       // zaman dilimi
   ulong            rates_mask,   // istenen serileri belirtmek için bayrak kombinasyo
   ulong            start,        // kopyalamanın başlayacağı başlangıç çubuğunun inde
   ulong            count         // kopyalanacak veri miktarı
   );
```

Başlangıç tarihine ve gerekli eleman sayısına göre erişim

```
bool  CopyRates(
   string           symbol,       // sembol adı
   ENUM_TIMEFRAMES  period,       // zaman dilimi
   ulong            rates_mask,   // istenen serileri belirtmek için bayrak kombinasyo
   datetime         from,         // başlangıç tarihi
   ulong            count         // kopyalanacak veri miktarı
   );
```

Gerekli tarih aralığının başlangıç ve bitiş tarihlerine göre erişim

```
bool  CopyRates(
   string           symbol,       // sembol adı
   ENUM_TIMEFRAMES  period,       // zaman dilimi
   ulong            rates_mask,   // istenen serileri belirtmek için bayrak kombinasyo
   datetime         from,         // başlangıç tarihi
   datetime         to            // bitiş tarihi
   );
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman dilimi.

rates_mask

[in] ENUM_COPY_RATES numaralandırmasından, istenen serilerin türünü belirten bayrak kombinasyonu. Verileri vektöre kopyalarken, ENUM_COPY_RATES numaralandırmasından yalnızca bir değer belirtilebilir, aksi takdirde hata meydana gelecektir.

start

[in] Kopyalanacak ilk elemanın indeksi.

count

[in] Kopyalanacak eleman sayısı.

from

[in] İlk elemana karşılık gelen çubuk zamanı.

to

[in] Son elemana karşılık gelen çubuk zamanı.

Geri dönüş değeri

Başarılı olursa true, hata olması durumunda false geri döndürür.

Not

Talep edilen veri aralığı sunucuda bulunan verileri aşıyorsa, fonksiyon false geri döndürür. Talep edilen veri aralığı TERMINAL_MAXBARS'ı (grafikteki maksimum çubuk sayısı) aşıyorsa, fonksiyon yine false geri döndürür.

Uzman Danışmandan veya komut dosyasından veri talep edilirken, terminalde yerel olarak ilgili veriler yoksa, sunucudan indirme işlemi başlatılır ve bu süreçte yerel geçmişten oluşturulabiliyorsa gerekli seriler oluşturulur. Fonksiyon, zaman aşımı süresi dolduğunda hazır olan veri miktarını geri döndürür, ancak geçmişin indirilme işlemi halen devam eder, böylece bir sonraki benzer talepte daha fazla veri geri döndürülür.

Başlangıç tarihine ve gerekli eleman sayısına göre veri talep edilirken, yalnızca belirtilen tarihe eşit veya bu tarihten önceki veriler geri döndürülür. Veri aralığı saniye hassasiyetiyle ayarlanır. Başka bir deyişle, değerin (hacim, makas, Açılış, Yüksek, Düşük, Kapanış veya Zaman) geri döndürüldüğü herhangi bir çubuğun açılış tarihi her zaman belirtilen tarihe eşit veya bu tarihten öncedir.

Belirli tarih aralığındaki veriler talep edilirken, yalnızca istenen aralıktaki veriler geri döndürülür. Veri aralığı saniye hassasiyetiyle ayarlanır. Başka bir deyişle, değerin (hacim, makas, Açılış, Yüksek, Düşük, Kapanış veya Zaman) geri döndürüldüğü herhangi bir çubuğun açılış tarihi her zaman istenen aralıkta bulunur.

Örneğin, haftanın mevcut günü Cumartesi ise, *start_time*=last_tuesday ve *stop_time*=last_friday ayarıyla haftalık zaman diliminde veri kopyalamaya çalışırken fonksiyon 0 geri döndürecektir, çünkü haftalık zaman diliminin açılış zamanı her zaman pazar gününe denk gelir ve dolayısıyla istenen aralığa hiçbir haftalık çubuk girmemektedir.

Mevcut tamamlanmamış çubuktan bir değer almanız gerekiyorsa, *start_pos*=0 ve *count*=1 ile ilk çağrı formunu kullanabilirsiniz.

ENUM_COPY_RATES

ENUM_COPY_RATES numaralandırması, matrise veya diziye aktarılacak verilerin türünü belirtmek için bayraklar içerir. Bayrak kombinasyonu, geçmişten birkaç serinin tek bir talepte alınmasına olanak tanır. Matristeki satırların sırası, ENUM_COPY_RATES numaralandırmasındaki değerlerin sırasına karşılık gelecektir. Diğer bir deyişle, Yüksek verileri içeren satır matriste her zaman Düşük verileri içeren satırdan daha yukarıda olacaktır.

Kimlik	Değer	Açıklama
COPY_RATES_OPEN	1	Açılış fiyatı serisi
COPY_RATES_HIGH	2	Yüksek fiyatı serisi
COPY_RATES_LOW	4	Düşük fiyatı serisi
COPY_RATES_CLOSE	8	Kapanış fiyatı serisi
COPY_RATES_TIME	16	Zaman serisi (çubuk açılış zamanı) Zaman serisinin float türündeki vektörlere ve matrislere (vectorf ve matrixf) alınması, float türünün hassasiyeti ciddi bir şekilde sınırlı olduğundan ve 1<<24'ten büyük int türü değerler float türünde doğru bir şekilde temsil edilemediğinden ~100 saniyelik kayıplara neden olur.
COPY_RATES_VOLUME_TICK	32	Tik hacim serisi
COPY_RATES_VOLUME_REAL	64	Gerçek hacim serisi
COPY_RATES_SPREAD	128	Makas serisi
Kombinasyon		
COPY_RATES_OHLC	15	Açılış, Yüksek, Düşük ve Kapanış serileri
COPY_RATES_OHLCT	31	Açılış, Yüksek, Düşük, Kapanış ve Zaman serileri

Örnek:

```
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- matrise fiyatlar al
  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. Hata ",GetLastError());
//--- kontrol et
  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). Hata ",GetLastError()
```

```
//--- vektöre fiyatlar al = geçersiz çağrı
  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. Hata ",GetLastError());
//--- vektöre kapanış fiyatları al
  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. Hata ",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(
    [1662436800,1662440400,1662444000,1662447600,1662451200,1662454800,1662458400,1662
   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,
*/
```

Ayrıca bakınız

Zaman Serilerine ve Göstergelere Erişim, CopyRates

CopyTicks

MqlTick yapısından matrise veya vektöre tikler alır. Elemanlar geçmişten şimdiki zamana doğru sayılır, yani indeksi 0 olan tik en eski tiktir. Tik analizi için, tikte tam olarak nelerin değiştiğini ifade eden *flags* alanını kontrol edin.

```
bool  matrix::CopyTicks(
   string           symbol,                 // sembol adı
   ulong            ticks_mask,             // alınacak tiklerin türünü belirten maske
   uint             flags=COPY_TICKS_ALL,   // alınacak tiklerin türünü belirten bayra
   ulong            from_msc=0,             // talep edilecek tiklerin başlangıç zaman
   ulong            count=0                 // alınacak tik sayısı
   );
```

Vektör metodu

```
bool  vector::CopyTicks(
   string           symbol,                 // sembol adı
   ulong            ticks_mask,             // alınacak tiklerin türünü belirten maske
   uint             flags=COPY_TICKS_ALL,   // alınacak tiklerin türünü belirten bayra
   ulong            from_msc=0,             // talep edilecek tiklerin başlangıç zaman
   ulong            count=0                 // alınacak tik sayısı
   );
```

Parametreler

symbol

[in] Sembol.

ticks_mask

[in] ENUM_COPY_TICKS numaralandırmasından, istenen tiklerin türünü belirten bayrak kombinasyonu. Verileri vektöre kopyalarken, ENUM_COPY_TICKS numaralandırmasından yalnızca bir değer belirtilebilir, aksi takdirde hata meydana gelecektir.

flags

[in] Alınacak tiklerin türünü belirten bayrak. COPY_TICKS_INFO - Alış ve/veya Satış fiyatı değişimleri sonucu oluşan tikler, COPY_TICKS_TRADE - Son ve/veya Hacim değişimleri sonucu oluşan tikler, COPY_TICKS_ALL - tüm tikler. Her istek türünde, MqlTick yapısının kalan alanlarına önceki tik değerleri eklenir.

from_msc

[in] Talep edilecek tiklerin başlangıç tarihi. Zaman, 01/01/1970 tarihinden itibaren olacak şekilde milisaniye cinsinden belirtilir. Eğer from=0 ise, count sayısı kadar son tikler geri döndürülür.

count

[in] Talep edilen tik sayısı. from_msc ve count parametreleri belirtilmezse, 2000'den fazla olmamak kaydıyla son tikler yazılacaktır.

Geri dönüş değeri

Başarılı olursa true, hata olması durumunda false geri döndürür.

Not

CopyTicks()'in ilk çağrısı, ilgili sembolün sabit diskte depolanan tik veri tabanının senkronizasyonunu başlatır. Yerel veri tabanı talep edilen tüm tikleri sağlayamıyorsa, eksik tikler otomatik olarak işlem sunucusundan indirilecektir. Bu durumda, CopyTicks()'te belirtilen ***from_msc*** zamanından mevcut zamana kadar olan tikler senkronize edilecektir. Bu sembol için sonrasında gelecek tüm tikler, tik veri tabanına eklenerek senkronize durumda tutulacaktır.

from_msc ve *count* parametreleri belirtilmezse, 2000'den fazla olmamak kaydıyla son tikler matrise/vektöre yazılacaktır.

Göstergelerde, CopyTicks() metodu sonucu hemen geri döndürür: bir göstergeden çağrıldığında, CopyTick() sembolün tüm kullanılabilir tiklerini hemen geri döndürür ve mevcut veri yeterli değilse tik veri tabanının senkronizasyonunu başlatır. Aynı sembol üzerindeki tüm göstergeler tek bir ortak iş parçacığında çalışır, bu nedenle gösterge senkronizasyonun tamamlanmasını bekleyemez. Senkronizasyon tamamlandıktan sonra, CopyTicks() bir sonraki çağrı sırasında talep edilen tüm tikleri geri döndürecektir. Göstergelerde, OnCalculate() fonksiyonu her tikin gelmesinden sonra çağrılır.

Uzman Danışmanlarda ve komut dosyalarında, CopyTicks() sonucu 45 saniye bekleyebilir: göstergelerden farklı olarak, her Uzman Danışman veya komut dosyası ayrı bir iş parçacığında çalışır ve bu nedenle senkronizasyonun tamamlanması için 45 saniyeye kadar bekleyebilir. Bu süre içerisinde gerekli sayıda tik sekronize edilemezse, CopyTicks() zaman aşımıyla kullanılabilir tikleri geri döndürecek ve senkronizasyona devam edecektir. Uzman Danışmanlarda OnTick(), her tikin işleyicisi değildir, sadece Uzman Danışmanı piyasadaki değişiklikler hakkında bilgilendirir. Bu, bir değişiklik grubu olabilir: terminal aynı anda birden fazla tiki alabilirken, OnTick() yalnızca bir kez çağrılarak Uzman Danışmanı en son piyasa durumu hakkında bilgilendirir.

Veri geri dönüş hızı: terminal, hızlı erişim önbelleğinde her enstrüman için 4096 son tiki depolar (Piyasa Derinliği çalışırken 65536 tik). Bu verilere ilişkin talepler en hızlı şekilde yürütülür. Geçerli işlem seansı için talep edilen tikler önbelleğin dışındaysa, CopyTicks() terminal belleğinde depolanan tikleri çağırır. Bu taleplerin tamamlanması daha fazla zaman gerektirir. En yavaş olan talepler diğer günlerdeki tiklerin talepleridir, çünkü bu durumda veriler diskten okunacaktır.

ENUM_COPY_TICKS

ENUM_COPY_TICKS numaralandırması, matrise veya diziye aktarılacak verilerin türünü belirtmek için bayraklar içerir. Bayrak kombinasyonu, geçmişten birkaç serinin tek bir talepte alınmasına olanak tanır. Matristeki satırların sırası, ENUM_COPY_TICKS numaralandırmasındaki değerlerin sırasına karşılık gelecektir. Diğer bir deyişle, Yüksek verileri içeren satır matriste her zaman Düşük verileri içeren satırdan daha yukarıda olacaktır.

Kimlik	Değer	Açıklama
COPY_TICKS_TIME_MS	1	Milisaniye cinsinden tik zamanı
COPY_TICKS_BID	2	Satış fiyatı
COPY_TICKS_ASK	4	Alış fiyatı
COPY_TICKS_LAST	8	Son fiyatı (son işlem fiyatı)
COPY_TICKS_VOLUME	16	Hacim

Kimlik	Değer	Açıklama
COPY_TICKS_FLAGS	32	Tik bayrakları

Tik analizi için, tikte tam olarak nelerin değiştiğini ifade eden bayrakları kullanın:

- TICK_FLAG_BID - tik, Satış fiyatı değişimi sonucu oluştu
- TICK_FLAG_ASK - tik, Alış fiyatı değişimi sonucu oluştu
- TICK_FLAG_LAST - tik, Son işlem fiyatı değişimi sonucu oluştu
- TICK_FLAG_VOLUME - tik, hacim değişimi sonucu oluştu
- TICK_FLAG_BUY - tik, alış işlemi sonucu oluştu
- TICK_FLAG_SELL - tik, satış işlemi sonucu oluştu

Ayrıca bakınız

Zaman Serilerine ve Göstergelere Erişim, CopyTicks

CopyTicksRange

Belirtilen tarih aralığı için MqlTick yapısından matrise veya vektöre tikler alır. Elemanlar geçmişten şimdiki zamana doğru sayılır, yani indeksi 0 olan tik en eski tiktir. Tik analizi için, tikte tam olarak nelerin değiştiğini ifade eden *flags* alanını kontrol edin.

```
bool  matrix::CopyTicksRange(
   string           symbol,                 // sembol adı
   ulong            ticks_mask,             // alınacak tiklerin türünü belirten maske
   uint             flags=COPY_TICKS_ALL,   // alınacak tiklerin türünü belirten bayra
   ulong            from_msc=0,             // talep edilecek tiklerin başlangıç zaman
   ulong            to_msc=0                // talep edilecek tiklerin bitiş zamanı
   );
```

Vektör metodu

```
bool  vector::CopyTicksRange(
   string           symbol,                 // sembol adı
   ulong            ticks_mask,             // alınacak tiklerin türünü belirten maske
   uint             flags=COPY_TICKS_ALL,   // alınacak tiklerin türünü belirten bayra
   ulong            from_msc=0,             // talep edilecek tiklerin başlangıç zaman
   ulong            to_msc=0                // talep edilecek tiklerin bitiş zamanı
   );
```

Parametreler

symbol

[in] Sembol.

ticks_mask

[in] ENUM_COPY_TICKS numaralandırmasından, istenen tiklerin türünü belirten bayrak kombinasyonu. Verileri vektöre kopyalarken, ENUM_COPY_TICKS numaralandırmasından yalnızca bir değer belirtilebilir, aksi takdirde hata meydana gelecektir.

flags

[in] Alınacak tiklerin türünü belirten bayrak. COPY_TICKS_INFO - Alış ve/veya Satış fiyatı değişimleri sonucu oluşan tikler, COPY_TICKS_TRADE - Son ve/veya Hacim değişimleri sonucu oluşan tikler, COPY_TICKS_ALL - tüm tikler. Her istek türünde, MqlTick yapısının kalan alanlarına önceki tik değerleri eklenir.

from_msc

[in] Talep edilecek tiklerin başlangıç tarihi. Zaman, 01/01/1970 tarihinden itibaren olacak şekilde milisaniye cinsinden belirtilir. from_msc parametresi belirtilmezse, geçmişin başından itibaren tikler geri döndürülür. Zaman >= from_msc olan tikler geri döndürülür.

to_msc

[in] Tiklerin talep edileceği aralığın bitiş zamanı. Zaman, 01/01/1970 tarihinden itibaren olacak şekilde milisaniye cinsinden belirtilir. Zaman <= from_msc olan tikler geri döndürülür. to_msc parametresi belirtilmezse, geçmişin sonuna kadar olan tüm tikler geri döndürülür.

Geri dönüş değeri

Başarılı olursa true, hata olması durumunda false geri döndürür. GetLastError() aşağıdaki hataları geri döndürebilir:

- ERR_HISTORY_TIMEOUT - tik senkronizasyonu için zaman aşımı meydana geldi, fonksiyon kullanılabilir tüm tikleri geri döndürdü.
- ERR_HISTORY_SMALL_BUFFER - statik arabellek çok küçük. Yalnızca dizinin depolayabileceği veri miktarı geri döndürüldü.
- ERR_NOT_ENOUGH_MEMORY - belirtilen aralıktan geçmiş verileri dinamik tik dizisine almak için yeterli bellek yok. Tik dizisi için yeterli bellek tahsis edilemedi.

Tik analizi için, tikte tam olarak nelerin değiştiğini ifade eden bayrakları kullanın:

- TICK_FLAG_BID - tik, Satış fiyatı değişimi sonucu oluştu
- TICK_FLAG_ASK - tik, Alış fiyatı değişimi sonucu oluştu
- TICK_FLAG_LAST - tik, Son işlem fiyatı değişimi sonucu oluştu
- TICK_FLAG_VOLUME - tik, hacim değişimi sonucu oluştu
- TICK_FLAG_BUY - tik, alış işlemi sonucu oluştu
- TICK_FLAG_SELL - tik, satış işlemi sonucu oluştu

Not

CopyTicksRange() metodu, tam olarak belirtilen aralıktan tikleri talep etmek için kullanılır. Örneğin, geçmişteki belirli bir gündeki tikler. CopyTicks() ise yalnızca başlangıç tarihinin belirtilmesine olanak sağlar, örneğin, ayın başından bugüne kadar olan tüm tikler.

Ayrıca bakınız

Zaman Serilerine ve Göstergelere Erişim, CopyTicksRange

Eye

Bir statik fonksiyondur. Köşegende 1'lerin ve diğer yerlerde 0'ların olduğu, istenilen büyüklükte bir matris oluşturur ve geri döndürür.

```
static matrix matrix::Eye(
  const ulong  rows,         // satır sayısı
  const ulong  cols,         // sütun sayısı
  const int    ndiag=0       // köşegenin indeksi
   );
```

Parametreler

rows

[in] Çıktıda satır sayısı.

cols

[in] Çıktıda sütun sayısı.

ndiag=0

[in] Köşegenin indeksi: ndiag=0 (varsayılan) ana köşegeni, ndiag>0 ana köşegenin üzerindeki köşegenleri ve ndiag<0 ana köşegenin altındaki köşegenleri ifade eder.

Geri dönüş değeri

Değerleri bire eşit olan k'nıncı köşegen dışında tüm elemanların sıfıra eşit olduğu bir matris.

MQL5 örneği:

```
  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]]
  */
```

Python örneği:

```
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

Bir statik fonksiyondur. Belirtilen büyüklükte (mutlaka kare olması gerekmez) bir birim matris oluşturur ve geri döndürür. Birim matris, ana köşegende 1'ler ve diğer yerlerde 0'lar içerir. Ana köşegen, [0,0],[1,1],[2,2] gibi satır ve sütun indeksleri eşit olan matris elemanlarından oluşur.

Ayrıca halihazırda mevcut olan bir matrisi bir birim matrise dönüştüren Identity metodu da vardır.

```
static matrix matrix::Identity(
  const ulong  rows,         // satır sayısı
  const ulong  cols,         // sütun sayısı
   );

void matrix::Identity();
```

Parametreler

rows

[in] n x n matristeki satır (ve sütun) sayısı.

Geri dönüş değeri

Bir birim matris geri döndürür. Birim matris, ana köşegende 1'lerin ve diğer yerlerde 0'ların olduğu bir kare matristir.

MQL5 örneği:

```
  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]]
*/
```

Python örneği:

```
np.identity(3)
array([[1.,  0.,  0.],
```

```
        [0.,  1.,  0.],
        [0.,  0.,  1.]])
```

Ones

Bir statik fonksiyondur. 1'lerle dolu yeni bir matris oluşturur ve geri döndürür.

```
static matrix matrix::Ones(
  const ulong  rows,      // satır sayısı
  const ulong  cols       // sütun sayısı
   );

static vector vector::Ones(
  const ulong  size,      // vektör büyüklüğü
   );
```

Parametreler

rows

[in] Satır sayısı.

cols

[in] Sütun sayısı.

Geri dönüş değeri

1'lerle dolu ve istenilen satır ve sütun sayısına sahip yeni bir matris.

MQL5 örneği:

```
  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]]
*/
```

Python örneği:

```
np.ones((4, 1))
array([[1.],
       [1.]])
```

Zeros

Bir statik fonksiyondur. 0'larla dolu yeni bir matris oluşturur ve geri döndürür.

```
static matrix matrix::Zeros(
  const ulong  rows,      // satır sayısı
  const ulong  cols       // sütun sayısı
   );

static vector vector::Zeros(
  const ulong  size,      // vektör büyüklüğü
   );
```

Parametreler

rows

[in] Satır sayısı.

cols

[in] Sütun sayısı.

Geri dönüş değeri

0'larla dolu ve istenilen satır ve sütun sayısına sahip yeni bir matris.

MQL5 örneği:

```
  matrix zeros=matrix::Zeros(3, 4);
  Print("zeros = \n", zeros);
/*
zeros =
   [[0,0,0,0]
    [0,0,0,0]
    [0,0,0,0]]
*/
```

Python örneği:

```
np.zeros((2, 1))
array([[ 0.],
       [ 0.]])
```

Full

Bir statik fonksiyondur. Belirtilen değerle dolu yeni bir matris oluşturur ve geri döndürür.

```
static matrix matrix::Full(
  const ulong   rows,       // satır sayısı
  const ulong   cols,       // sütun sayısı
  const double  value       // doldurulacak değer
   );

static vector vector::Full(
  const ulong   size,       // vektör büyüklüğü
  const double  value       // doldurulacak değer
   );
```

Parametreler

rows

[in] Satır sayısı.

cols

[in] Sütun sayısı.

value

[in] Tüm matris elemanları için doldurulacak değer.

Geri dönüş değeri

Belirtilen değerle dolu ve istenilen satır ve sütun sayısına sahip yeni bir matris geri döndürür.

MQL5 örneği:

```
  matrix full=matrix::Full(3,4,10);
  Print("full = \n", full);
/*
full =
   [[10,10,10,10]
    [10,10,10,10]
    [10,10,10,10]]
*/
```

Örnek:

```
np.full((2, 2), 10)
array([[10, 10],
       [10, 10]])
```

Tri

Bir statik fonksiyondur. İstenilen köşegende ve altında 1'lerin ve diğer yerlerde 0'ların olduğu bir matris oluşturur ve geri döndürür.

```
static matrix matrix::Tri(
  const ulong rows,         // satır sayısı
  const ulong cols,         // sütun sayısı
  const int   ndiag=0       // köşegenin indeksi
   );
```

Parametreler

rows

[in] Satır sayısı.

cols

[in] Sütun sayısı.

ndiag=0

[in] Köşegenin indeksi: ndiag=0 (varsayılan) ana köşegeni, ndiag>0 ana köşegenin üzerindeki köşegenleri ve ndiag<0 ana köşegenin altındaki köşegenleri ifade eder.

Geri dönüş değeri

İstenilen köşegende ve altında 1'lerin ve diğer yerlerde 0'ların olduğu bir matris.

MQL5 örneği:

```
   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]]
*/
```

Örnek:

```
np.tri(3, 5, 2, dtype=int)
array([[1, 1, 1, 0, 0],
       [1, 1, 1, 1, 0],
       [1, 1, 1, 1, 1]])
```

Init

Matrisi veya vektörü başlatır.

```
void matrix::Init(
  const ulong  rows,            // satır sayısı
  const ulong  cols,            // sütun sayısı
  func_name    init_func=NULL,  // sınıfın bir kapsamına veya statik metoduna yerleşti
   ...         parameters
   );

void vector::Init(
  const ulong  size,            // vektör büyüklüğü
  func_name    init_func=NULL,  // sınıfın bir kapsamına veya statik metoduna yerleşti
   ...         parameters
   );
```

Parametreler

rows

[in] Satır sayısı.

cols

[in] Sütun sayısı.

func_name

[in] Başlatma fonksiyonu.

...

[in] Başlatma fonksiyonunun parametreleri.

Geri dönüş değeri

Geri dönüş değeri yoktur.

Örnek:

```
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;
  }
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
  int size_m=3, size_k=4;
  matrix  m(size_m,size_k,MatrixArange,-2.,0.1); // önce size_m x size_k büyüklüğünde
  Print("matrix m \n",m);                        // daha sonra başlatma sırasında list
  matrixf m_float(5,5,MatrixArange,-2.f,0.1f);   // float türünde bir matris oluşturul
  Print("matrix m_float \n",m_float);
  vector  v(size_k,VectorArange,-10.0);          // vektör oluşturulduktan sonra, tek
  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

Matrisi veya vektörü belirtilen değerle doldurur.

```
void matrix::Fill(
  const double  value      // doldurulacak değer
   );

void vector::Fill(
  const double  value      // doldurulacak değer
   );
```

Parametreler

value

[in] Tüm matris elemanları için doldurulacak değer.

Geri dönüş değeri

Geri dönüş değeri yoktur. Matris belirtilen değerle doldurulur.

Örnek:

```
   matrix matrix_a(2,2);
   matrix_a.Fill(10);
   Print("matrix_a\n",matrix_a);

/*
  matrix_a
  [[10,10]
   [10,10]]
*/
```

Matris ve Vektör Manipülasyonları

Bunlar, temel matris işlemleri için metotlardır: doldurma, kopyalama, matrisin bir parçasını alma, devriğini alma, bölme ve sıralama.

Matris satırları ve sütunlarıyla işlemler için de çeşitli metotlar bulunmaktadır.

Fonksiyon	Eylem
HasNan	Matris/vektördeki NaN değerlerinin sayısını geri döndürür
Transpose	Matrisin devriğini alır (eksenleri değiştirir) ve değiştirilmiş matrisi geri döndürür
TriL	k'ıncı köşegenin üzerindeki elemanların 0'a çevrildiği matrisin bir kopyasını geri döndürür. Alt üçgen matris
TriU	k'ıncı köşegenin altındaki elemanların 0'a çevrildiği matrisin bir kopyasını geri döndürür. Üst üçgen matris
Diag	Bir köşegeni çıkarır veya bir köşegen matris oluşturur
Row	Bir satır vektörü geri döndürür. Belirtilen satıra vektörü yazar
Col	Bir sütun vektörü geri döndürür. Belirtilen sütuna vektörü yazar
Copy	Matrisin/vektörün bir kopyasını geri döndürür
Compare	Belirtilen hassasiyetle iki matrisin/vektörün elemanlarını karşılaştırır
CompareByDigits	Anlamlı basamak hassasiyetiyle iki matrisin/vektörün elemanlarını karşılaştırır
Flat	Matris elemanının iki yerine bir indeks aracılığıyla adreslenmesine olanak sağlar
Clip	Matrisin/vektörün elemanlarını belirli bir geçerli değerler aralığıyla sınırlar
Reshape	Verilerini değiştirmeden matrisin şeklini değiştirir
Resize	Belirtilen şekle ve büyüklüğe sahip yeni bir matris geri döndürür
SwapRows	Matristeki satırları değiştirir
SwapCols	Matristeki sütunları değiştirir
Split	Bir matrisi birden çok alt matrise böler
Hsplit	Bir matrisi yatay olarak birden çok alt matrise böler. axis=0 Split ile aynıdır
Vsplit	Bir matrisi dikey olarak birden çok alt matrise böler. axis=1 Split ile aynıdır
ArgSort	Matris veya vektörü dolaylı sıralar
Sort	Matris veya vektörü yerinde sıralar

HasNan

Matris/vektördeki NaN değerlerinin sayısını geri döndürür.

```
ulong vector::HasNan();

ulong matrix::HasNan();
```

Geri dönüş değeri

NaN değerini içeren matris/vektör elemanlarının sayısı.

Not

NaN değerlerine sahip uygun eleman çiftini karşılaştırırken Compare ve CompareByDigits metotları bu elemanları eşit kabul ederken, kesirli sayıların olağan karşılaştırması durumunda NaN != NaN olur.

Örnek:

```
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);
  }

/* Sonuç:

 single: false
 vector: true
*/
```

Ayrıca bakınız

MathClassify, Compare, CompareByDigits

Transpose

Matrisin devriğini alır (eksenleri değiştirir) ve değiştirilmiş matrisi geri döndürür.

```
matrix matrix::Transpose()
```

Geri dönüş değeri

Devrik matris.

MQL5'te matrisin devriğini almak için basit bir algoritma:

```
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);
  }
```

MQL5 örneği:

```
  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]]
  */
```

Python örneği:

```
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

k'ıncı köşegenin üzerindeki elemanların 0'a çevrildiği matrisin bir kopyasını geri döndürür. Alt üçgen matris.

```
matrix matrix::Tril(
  const int      ndiag=0      // köşegenin indeksi
   );
```

Parametreler

ndiag=0

[in] Üzerindeki elemanların sıfırlarla değiştirileceği köşegen. ndiag=0 (varsayılan) ana köşegeni, ndiag>0 ana köşegenin üzerindeki köşegenleri ve ndiag<0 ana köşegenin altındaki köşegenleri ifade eder.

Geri dönüş değeri

İstenilen köşegende ve altında 1'lerin ve diğer yerlerde 0'ların olduğu bir matris.

MQL5 örneği:

```
   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]]
  */
```

Python örneği:

```
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

k'ıncı köşegenin altındaki elemanların 0'a çevrildiği matrisin bir kopyasını geri döndürür. Üst üçgen matris.

```
matrix matrix::Triu(
  const int      ndiag=0      // köşegenin indeksi
   );
```

Parametreler

ndiag=0

[in] Altındaki elemanların sıfırlarla değiştirileceği köşegen. ndiag=0 (varsayılan) ana köşegeni, ndiag>0 ana köşegenin üzerindeki köşegenleri ve ndiag<0 ana köşegenin altındaki köşegenleri ifade eder.

MQL5 örneği:

```
   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]]
  */
```

Python örneği:

```
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

Bir köşegeni çıkarır veya bir köşegen matris oluşturur.

```
vector matrix::Diag(
  const int     ndiag=0      // köşegenin indeksi
   );

void matrix::Diag(
  const vector  v,           // köşegene yazılacak vektör
  const int     ndiag=0      // köşegenin indeksi
   );
```

Parametreler

v

[in] Elemanları ilgili köşegene (ndiag=0 ana köşegendir) yazılacak vektör.

ndiag=0

[in] Köşegenin indeksi: ndiag=0 (varsayılan) ana köşegeni, ndiag>0 ana köşegenin üzerindeki köşegenleri ve ndiag<0 ana köşegenin altındaki köşegenleri ifade eder.

Not

Büyüklüğü belirlenmemiş matrisler için köşegen ayarlanabilir. Bu durumda, köşegene yazılacak vektörün büyüklüğünde bir sıfır matrisi oluşturulacak ve vektör elemanları ilgili köşegene doldurulacaktır. Köşegen halihazırda mevcut olan bir matrise ayarlanırsa, matrisin büyüklüğü ve köşegene yazılacak vektör dışındaki elemanları değişmez.

Örnek:

```
   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

Bir satır vektörü geri döndürür. Belirtilen satıra vektörü yazar

```
vector matrix::Row(
  const ulong   nrow      // satırın indeksi
   );

void matrix::Row(
  const vector  v,        // satıra yazılacak vektör
  const ulong   nrow      // satırın indeksi
   );
```

Parametreler

nrow

[in] Satırın indeksi.

Geri dönüş değeri

Vektör.

Not

Büyüklüğü belirlenmemiş matrisler için satır ayarlanabilir. Bu durumda, satıra yazılacak vektör büyüklüğü x satırın indeksi+1 büyüklüğünde bir sıfır matrisi oluşturulacak ve vektör elemanları ilgili satıra doldurulacaktır. Satır halihazırda mevcut olan bir matrise ayarlanırsa, matrisin büyüklüğü ve satıra yazılacak vektör dışındaki elemanları değişmez.

Örnek:

```
   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

Bir sütun vektörü geri döndürür. Belirtilen sütuna vektörü yazar

```
vector matrix::Col(
  const ulong   ncol      // sütunun indeksi
   );

void matrix::Col(
  const vector  v,        // sütuna yazılacak vektör
  const ulong   ncol      // sütunun indeksi
   );
```

Parametreler

ncol

[in] Sütunun indeksi.

Geri dönüş değeri

Vektör.

Not

Büyüklüğü belirlenmemiş matrisler için sütun ayarlanabilir. Bu durumda, sütuna yazılacak vektör büyüklüğü x sütunun indeksi+1 büyüklüğünde bir sıfır matrisi oluşturulacak ve vektör elemanları ilgili sütuna doldurulacaktır. Sütun halihazırda mevcut olan bir matrise ayarlanırsa, matrisin büyüklüğü ve sütuna yazılacak vektör dışındaki elemanları değişmez.

Örnek:

```
   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

Matrisin/vektörün bir kopyasını oluşturur.

```
bool matrix::Copy(
  const matrix&  a     // kopyalanacak matris
   );
bool vector::Copy(
  const vector&  v     // kopyalanacak vektör
   );
```

Parametreler

v

[in] Kopyalanacak matris veya vektör.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

MQL5 örneği:

```
  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]]
  */
  */
```

Python örneği:

```
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

Belirtilen hassasiyetle iki matrisin/vektörün elemanlarını karşılaştırır.

```
ulong vector::Compare(
  const vector& vec,          // karşılaştırılacak vektör
  const double  epsilon       // hassasiyet
   );

ulong matrix::Compare(
  const matrix& mat,          // karşılaştırılacak matris
  const double  epsilon       // hassasiyet
   );
```

Parametreler

vector_b

[in] Karşılaştırılacak vektör.

epsilon

[in] Hassasiyet.

Geri dönüş değeri

Karşılaştırılan matrislerin veya vektörlerin eşleşmeyen elemanlarının sayısı: matrisler eşitse 0, aksi takdirde 0'dan büyük olacaktır.

Not

== veya != karşılaştırma operatörleri, eleman bazında tam karşılaştırma yürütür. Reel sayıların tam karşılaştırmasının sınırlı kullanımı olduğu bilindiğinden, epsilon karşılaştırma yöntemi eklenmiştir. Bir matris, örneğin 1e-20 ila 1e+20 aralığında değerlere sahip elemanlar içerebilir. Bu tür matrisler, anlamlı basamak hassasiyetiyle eleman bazında karşılaştırma kullanılarak işlenebilir.

Örnek:

```
   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

Anlamlı basamak hassasiyetiyle iki matrisin/vektörün elemanlarını karşılaştırır.

```
ulong vector::CompareByDigits(
  const vector& vec,          // karşılaştırılacak vektör
  const int     digits        // anlamlı basamak sayısı
   );

ulong matrix::CompareByDigits(
  const matrix& mat,          // karşılaştırılacak matris
  const int     digits        // anlamlı basamak sayısı
   );
```

Parametreler

vector_b

[in] Karşılaştırılacak vektör.

digits

[in] Karşılaştırılacak anlamlı basamakların sayısı.

epsilon

[in] Karşılaştırma hassasiyeti. Mutlak olarak iki değer belirtilen hassasiyetten daha az farklılık gösteriyorsa, eşit olarak kabul edilirler.

Geri dönüş değeri

Karşılaştırılan matrislerin veya vektörlerin eşleşmeyen elemanlarının sayısı: matrisler eşitse 0, aksi takdirde 0'dan büyük olacaktır.

Not

== veya != karşılaştırma operatörleri, eleman bazında tam karşılaştırma yürütür. Reel sayıların tam karşılaştırmasının sınırlı kullanımı olduğu bilindiğinden, epsilon karşılaştırma yöntemi eklenmiştir. Bir matris, örneğin 1e-20 ila 1e+20 aralığında değerlere sahip elemanlar içerebilir. Bu tür matrisler, anlamlı basamak hassasiyetiyle eleman bazında karşılaştırma kullanılarak işlenebilir.

Örnek:

```
   int    size_m=128;
   int    size_k=256;
   matrix matrix_a(size_m,size_k);
//--- matrisi doldur
   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;
           }
        }
     }
//--- başka bir matris al
   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

Matris elemanının iki yerine bir indeks aracılığıyla adreslenmesine olanak sağlar.

```
bool matrix::Flat(
  const ulong   index,      //
  const double  value       // ayarlanacak değer
   );

double matrix::Flat(
  const ulong   index,      //
   );
```

Parametreler

index

[in] Flat indeksi.

value

[in] Ayarlanacak değer.

Geri dönüş değeri

Belirtilen indekse karşılık gelen değer.

Not

mat(3,3) matrisi için erişim aşağıdaki gibi yazılabilir:

- okuma: x=mat.Flat(4), x=mat[1][1] ile eşdeğerdir
- yazma: mat.Flat(5, 42), mat[1][2]=42 ile eşdeğerdir

Örnek:

```
   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

Matrisin/vektörün elemanlarını belirli bir geçerli değerler aralığıyla sınırlar.

```
bool matrix::Clip(
  const double  min_value,      // minimum değer
  const double  max_value       // maksimum değer
   );
bool vector::Clip(
  const double  min_value,      // minimum değer
  const double  max_value       // maksimum değer
   );
```

Parametreler

min_value

[in] Minimum değer.

max_value

[in] Maksimum değer.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Not

Matris (veya vektör) yerinde işlenir. Kopya oluşturulmaz.

Örnek:

```
   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

Verilerini değiştirmeden matrisin şeklini değiştirir.

```
void  Reshape(
  const ulong  rows,      // yeni satır sayısı
  const ulong  cols       // yeni sütun sayısı
   );
```

Parametreler

rows

[in] Yeni satır sayısı.

cols

[in] Yeni sütun sayısı.

Not

Matris yerinde işlenir. Kopya oluşturulmaz. Herhangi bir büyüklük belirtilebilir, yani, rows_new*cols_new!=rows_old*cols_old. Matris arabelleği artırıldığında, ekstra değerler tanımsız olur.

Örnek:

```
   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

Belirtilen şekle ve büyüklüğe sahip yeni bir matris geri döndürür.

```
bool matrix::Resize(
  const ulong  rows,     // yeni satır sayısı
  const ulong  cols,     // yeni sütun sayısı
  const ulong  reserve=0 // korunacak eleman sayısı
   );

bool vector::Resize(
  const ulong  size,     // yeni büyüklük
  const ulong  reserve=0 // korunacak eleman sayısı
   );
```

Parametreler

rows

[in] Yeni satır sayısı.

cols

[in] Yeni sütun sayısı.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Not

Matris (veya vektör) yerinde işlenir. Kopya oluşturulmaz. Herhangi bir büyüklük belirtilebilir, yani, rows_new*cols_new!=rows_old*cols_old. Reshape'den farklı olarak, matris satır satır işlenir. Sütun sayısı artırıldığında, yeni sütunların elemanları tanımsız olur. Satır sayısı artırıldığında da yeni satırların elemanları tanımsız olur. Sütun sayısı azaltıldığında ise matrisin her bir satırı kesilir.

Örnek:

```
   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

Belirtilen indekse göre vektör elemanının değerini ayarlar.

```
bool vector::Set(
  ulong   index,      // elemanın indeksi
  double  value       // değer
   );
```

Parametreler

index

[in] Değerin ayarlanması gereken elemanın indeksi.

value

[in] Değer.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Not

Set metodu, köşeli parantez kullanarak bir değer atamakla aynı şeyi yapar, yani: *vektör[indeks] =değer*. Bu metot, bu tür bir gösterimin kullanıldığı dillerden kod aktarımını kolaylaştırmak için eklenmiştir. Aşağıdaki örnekte, vektörü belirtilen indekse göre değerlerle doldurmak için her iki seçenek de gösterilmektedir.

Örnek:

```
void OnStart()
  {
//---
   vector v1(10, VectorAssignValues);
   Print("v1 = ", v1);

   vector v2(10, VectorSetValues);
   Print("v2 = ", v2);
  }
 /* Sonuç
  v1 = [1,2,4,8,16,32,64,128,256,512]
  v2 = [1,2,4,8,16,32,64,128,256,512]
  */
//+------------------------------------------------------------------+
//| Atama işlemi aracılığıyla bir vektörü bir sayının kuvvetleriyle doldur  |
//+------------------------------------------------------------------+
void VectorAssignValues(vector& v, double initial=1)
  {
   double value=initial;
```

```
   for(ulong k=0; k<v.Size(); k++)
     {
      v[k]=value;
      value*=2;
     }
  }
//+------------------------------------------------------------------------------+
//| Set metodunu kullanarak bir vektörü bir sayının kuvvetleriyle doldur         |
//+------------------------------------------------------------------------------+
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

Matristeki satırları değiştirir.

```
bool matrix::SwapRows(
  const ulong  row1,     // ilk satırın indeksi
  const ulong  row2      // ikinci satırın indeksi
   );
```

Parametreler

row1

[in] İlk satırın indeksi.

row2

[in] İkinci satırın indeksi.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Örnek:

```
   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

Matristeki sütunları değiştirir.

```
bool matrix::SwapCols(
  const ulong  row1,     // ilk sütunun indeksi
  const ulong  row2      // ikinci sütunun indeksi
   );
```

Parametreler

col1

[in] İlk sütunun indeksi.

col2

[in] İkinci sütunun indeksi.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Örnek:

```
   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

Bir matrisi birden çok alt matrise böler.

```
bool matrix::Split(
  const ulong  parts,      // alt matris sayısı
  const int    axis,       // eksen
  matrix&      splitted[]  // ortaya çıkan alt matris
   );

void matrix::Split(
  const ulong& parts[],    // alt matrislerin büyüklükleri
  const int    axis,       // eksen
  matrix&      splitted[]  // ortaya çıkan alt matris
   );
```

Parametreler

parts

[in] Matrisin bölüneceği alt matris sayısı.

axis

[in] Eksen. 0 - yatay eksen, 1 - dikey eksen.

splitted

[out] Ortaya çıkan alt matris.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Not

Alt matris sayısı belirtilirse, aynı büyüklükte alt matrisler elde edilir. Bu, matris büyüklüğünün (0 - satır sayısı, 1 - sütun sayısı) parts parametresine kalansız olarak bölünebilir olması gerektiği anlamına gelir. Elde edilecek alt matrislerin büyüklükleri belirtilerek farklı büyüklüklerde alt matrisler elde edilebilir. Matris belirtilen büyüklükteki alt matrisler için bölünmeye başlayacaktır. Eğer belirtilen büyüklükteki alt matrisler dolduysa ve matris hala tamamen bölünmediyse, bölünmemiş kalan kısım son alt matris olarak yazılacaktır.

Örnek:

```
   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

Bir matrisi yatay olarak birden çok alt matrise böler. axis=0 Split ile aynıdır

```
bool matrix::Hsplit(
  const ulong  parts,      // alt matris sayısı
  matrix&      splitted[]  // ortaya çıkan alt matris
   );

void matrix::Hsplit(
  const ulong& parts[],    // alt matrislerin büyüklükleri
  matrix&      splitted[]  // ortaya çıkan alt matris
   );
```

Parametreler

parts

[in] Matrisin bölüneceği alt matris sayısı.

splitted

[out] Ortaya çıkan alt matris.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Not

Alt matris sayısı belirtilirse, aynı büyüklükte alt matrisler elde edilir. Bu, satır sayısının parts parametresine kalansız olarak bölünebilir olması gerektiği anlamına gelir. Elde edilecek alt matrislerin büyüklükleri belirtilerek farklı büyüklüklerde alt matrisler elde edilebilir. Matris belirtilen büyüklükteki alt matrisler için bölünmeye başlayacaktır. Eğer belirtilen büyüklükteki alt matrisler dolduysa ve matris hala tamamen bölünmediyse, bölünmemiş kalan kısım son alt matris olarak yazılacaktır.

Örnek:

```
   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

Bir matrisi dikey olarak birden çok alt matrise böler. axis=1 Split ile aynıdır

```
bool matrix::Vsplit(
  const ulong  parts,      // alt matris sayısı
  matrix&      splitted[]  // ortaya çıkan alt matris
   );

void matrix::Vsplit(
  const ulong& parts[],    // alt matrislerin büyüklükleri
  matrix&      splitted[]  // ortaya çıkan alt matris
   );
```

Parametreler

parts

[in] Matrisin bölüneceği alt matris sayısı.

splitted

[out] Ortaya çıkan alt matris.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Not

Alt matris sayısı belirtilirse, aynı büyüklükte alt matrisler elde edilir. Bu, sütun sayısının parts parametresine kalansız olarak bölünebilir olması gerektiği anlamına gelir. Elde edilecek alt matrislerin büyüklükleri belirtilerek farklı büyüklüklerde alt matrisler elde edilebilir. Matris belirtilen büyüklükteki alt matrisler için bölünmeye başlayacaktır. Eğer belirtilen büyüklükteki alt matrisler dolduysa ve matris hala tamamen bölünmediyse, bölünmemiş kalan kısım son alt matris olarak yazılacaktır.

Örnek:

```
   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

Matris veya vektörü dolaylı sıralar.

```
vector vector::Sort(
  func_name  compare_func=NULL,  // karşılaştırma fonksiyonu
  T          context             // özel sıralama fonksiyonu için parametre
   );

matrix matrix::Sort(
  func_name  compare_func=NULL   // karşılaştırma fonksiyonu
  T          context             // özel sıralama fonksiyonu için parametre
   );

matrix matrix::Sort(
  const int  axis,               // sıralama için eksen
  func_name  compare_func=NULL   // karşılaştırma fonksiyonu
  T          context             // özel sıralama fonksiyonu için parametre
   );
```

Parametreler

axis

[in] Sıralamanın gerçekleştirileceği eksen: 0 yatay, 1 dikey.

func_name

[in] Karşılaştırıcı. ENUM_SORT_MODE numaralandırma değerlerinden birini veya kendi karşılaştırma fonksiyonunuzu belirtebilirsiniz. Herhangi bir fonksiyon belirtilmezse, artan sıralama kullanılır.

Özel karşılaştırma fonksiyonu şu iki türde olabilir:

- int comparator(T x1,T x2)
- int comparator(T x1,T x2,TContext context)

Burada T, matris veya vektörün türüdür, TContex ise Sort metoduna ek bir parametre olarak iletilen context değişkeninin türüdür.

context

[in] Özel sıralama fonksiyonuna iletilebilen isteğe bağlı ek parametre.

Geri dönüş değeri

Sıralanmış elemanların indekslerini içeren vektör veya matris. Örneğin, [4,2,0,1,3] sonucu, sıfır konumunda indeksi 4 olan elemanın, birinci konumda indeksi 2 olan elemanın, ... olması gerektiğini ifade eder.

Sort

Matris veya vektörü yerinde sıralar

```
void vector::Sort(
  func_name  compare_func=NULL,  // karşılaştırma fonksiyonu
  T          context             // özel sıralama fonksiyonu için parametre
   );

void matrix::Sort(
  func_name  compare_func=NULL   // karşılaştırma fonksiyonu
  T          context             // özel sıralama fonksiyonu için parametre
   );

void matrix::Sort(
  const int  axis,               // sıralama için eksen
  func_name  compare_func=NULL   // karşılaştırma fonksiyonu
  T          context             // özel sıralama fonksiyonu için parametre
   );
```

Parametreler

axis

[in] Sıralamanın gerçekleştirileceği eksen: 0 yatay, 1 dikey.

func_name

[in] Karşılaştırıcı. ENUM_SORT_MODE numaralandırma değerlerinden birini veya kendi karşılaştırma fonksiyonunuzu belirtebilirsiniz. Herhangi bir fonksiyon belirtilmezse, artan sıralama kullanılır.

Özel karşılaştırma fonksiyonu şu iki türde olabilir:

- int comparator(T x1,T x2)
- int comparator(T x1,T x2,TContext context)

Burada T, matris veya vektörün türüdür, TContex ise Sort metoduna ek bir parametre olarak iletilen context değişkeninin türüdür.

context

[in] Özel sıralama fonksiyonuna iletilebilen isteğe bağlı ek parametre.

Geri dönüş değeri

Yok. Sıralama yerinde gerçekleştirilir, yani Sort metodunun çağrıldığı matris/vektörün verilerine uygulanır.

Örnek:

```
//+------------------------------------------------------------------+
//| Sıralama fonksiyonu                                              |
//+------------------------------------------------------------------+
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);
  }
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- vektörü doldur
   vector v(100);
   //--- artan düzende sırala
   v.Sort(MyDoubleComparator);   // burada varsayılan değeri 0 olan ek parametre kulla
   Print(v);
   // azalan düzende sırala
   v.Sort(MyDoubleComparator,1); // burada ek parametre olarak 1 değeri kullanıcı tara
   Print(v);
  }
```

Matrisler ve Vektörlerle Matematiksel İşlemler

Toplama, çıkarma, çarpma ve bölme gibi matematiksel işlemler, matrisler ve vektörler üzerinde eleman bazında gerçekleştirilebilir.

Matematiksel fonksiyonlar başlangıçta skaler değerler üzerinde ilgili işlemleri gerçekleştirmek için tasarlanmıştır. Fonksiyonların çoğu artık matrislere ve vektörlere uygulanabilmektedir. Bu fonksiyonlar şunlardır: MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh ve MathTanh. Bu tür işlemler sırasında, matrisler ve vektörler eleman bazında işlenir. Örnek:

```
//---
  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]]
  */
```

MathMod ve MathPow için ikinci eleman, uygun büyüklükte bir skaler veya matris/vektör olabilir.

Aşağıdaki örnek, bir vektöre matematiksel fonksiyonları uygulayarak standart sapmanın nasıl hesaplanacağını göstermektedir.

```
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- vektörü doldurmak için başlatma fonksiyonunu kullan
  vector r(10, ArrayRandom); // 0'dan 1'e rastgele sayılar dizisi
//--- ortalama değeri hesapla
  double avr=r.Mean();       // dizi ortalama değeri
  vector d=r-avr;            // ortalama değerden sapmalar dizisini hesapla
  Print("avr(r)=", avr);
  Print("r=", r);
  Print("d=", d);
  vector s2=MathPow(d, 2);   // sapmaların karelerinin dizisi
  double sum=s2.Sum();       // sapmaların karelerinin toplamı
//--- standart sapmayı iki farklı şekilde hesapla
  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
*/
//+------------------------------------------------------------------+
//| Vektörü rastgele değerlerle doldur                               |
//+------------------------------------------------------------------+
void ArrayRandom(vector& v)
 {
  for(ulong i=0; i<v.Size(); i++)
    v[i]=double(MathRand())/32767.;
 }
```

Matematiksel İşlemler

Toplama, çıkarma, çarpma ve bölme gibi matematiksel işlemler, matrisler ve vektörler üzerinde eleman bazında gerçekleştirilebilir.

Her iki matris veya her iki vektör aynı türde ve aynı büyüklükte olmalıdır. Matrisin her elemanı, ikinci matrisin karşılık gelen elemanıyla işleme girer.

İkinci eleman (çarpan, çıkaran veya bölen) olarak uygun türde (double, float veya complex) bir skaler de kullanılabilir. Bu durumda, matrisin veya vektörün her bir elemanı belirtilen skalerle işleme girecektir.

```
  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;    // Hadamard çarpımı, matrix çarpımı ile karışt
  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;

//--- yerinde işlemler mümkündür
  matrix_a+=matrix_b;
  matrix_a/=2;
```

Aynı işlemler vektörler için de mevcuttur.

Matematiksel Fonksiyonlar

Şu matematiksel fonksiyonlar matrislere ve vektörlere uygulanabilir: MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh ve MathTanh. Bu tür işlemler sırasında, matrisler ve vektörler eleman bazında işlenir.

MathMod ve MathPow için ikinci eleman, uygun büyüklükte bir skaler veya matris/vektör olabilir.

```
   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++;
     }
```

Matris ve Vektör Çarpımları

Matris ve vektör çarpımı hesaplamaları şunları içerir:

- Matris çarpımı
- Vektör çarpımı
- Kovaryans matrisinin hesaplanması
- İki vektörün çapraz korelasyonunun hesaplanması
- iki vektörün konvolüsyonunun hesaplanması
- Korelasyon katsayısının hesaplanması

Fonksiyon	Eylem
MatMul	İki matrisin matris çarpımı
GeMM	Genel matris çarpımı (General Matrix Multiplication, GeMM)
Power	Kare matrisi bir tam sayı kuvvete yükseltir
Dot	İki vektörün nokta çarpımı
Kron	İki matrisin, matrisin ve vektörün, vektörün ve matrisin veya iki vektörün Kronecker çarpımını geri döndürür
Inner	İki matrisin iç çarpımı
Outer	İki matrisin veya iki vektörün dış çarpımını hesaplar
CorrCoef	Pearson korelasyon katsayısını (lineer korelasyon katsayısı) hesaplar
Cov	Kovaryans matrisini hesaplar
Correlate	İki vektörün çapraz korelasyonunu hesaplar
Convolve	İki vektörün ayrık, lineer konvolüsyonunu geri döndürür

MatMul

Matrislerin ve vektörlerin çarpımını sağlayan MatMul metodu birkaç aşırı yüke sahiptir.

Bir matrisi bir matrisle çarpma: `matrix[M][K] * matrix[K][N] = matrix[M][N]`

```
matrix  matrix::MatMul(
  const matrix&  b      // ikinci matris
   );
```

Bir vektörü bir matrisle çarpma: `horizontal vector[K] * matrix[K][N] = horizontal vector[N]`

```
vector  vector::MatMul(
  const matrix&  b      // matris
   );
```

Bir matrisi bir vektörle çarpma: `matrix[M][K] * vertical vector[K] = vertical vector[M]`

```
vector  matrix::MatMul(
  const vector&  b      // vektör
   );
```

Skaler vektör çarpımı: `horizontal vector * vertical vector = dot value`

```
scalar  vector::MatMul(
  const vector&  b      // ikinci vektör
   );
```

Parametreler

b

[in] Matris veya vektör.

Geri dönüş değeri

Kullanılan metoda bağlı olarak matris, vektör veya skaler.

Not

Matrisler çarpma işlemi için uyumlu olmalıdır, yani birinci matristeki sütun sayısı, ikinci matristeki satır sayısına eşit olmalıdır. Matris çarpımı değişmeli değildir: genel olarak, birinci matrisin ikinci matrisle çarpımının sonucu, ikinci matrisin birinci matrisle çarpımının sonucuna eşit değildir.

Matris çarpımı, birinci matrisin satır vektörlerinin ve ikinci matrisin sütun vektörlerinin skaler çarpımlarının tüm olası kombinasyonlarından oluşur.

Skaler çarpımda vektörler aynı uzunlukta olmalıdır.

Bir vektör ile bir matris çarpılırken, vektörün uzunluğu matrisin sütun sayısıyla tam olarak eşleşmelidir.

MQL5'te saf matris çarpımı algoritması:

```
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);
  }
```

Matris çarpımı örneği:

```
   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]]
*/
```

Bir yatay vektörü bir matrisle çarpma örneği:

```
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
```

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 3x5 matris oluştur
   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));

  /* Sonuç
    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]
   */
  }
//+------------------------------------------------------------------+
//|  Matrisi artan değerlerle doldur                                 |
//+------------------------------------------------------------------+
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;
        }
     }
//---
  }
```

Bir matrisi bir dikey vektörle çarpma örneği:

```
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
```

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 3x5 matris oluştur
   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));

  /* Sonuç
   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]
   */
  }
//+------------------------------------------------------------------+
//|  Matrisi artan değerlerle doldur                                 |
//+------------------------------------------------------------------+
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;
        }
     }
//---
  }
```

Vektörlerin skaler (nokta) çarpımına bir örnek:

```
void OnStart()
  {
```

```
//--- bir yatay vektör ile bir dikey vektörün skaler çarpımı
   vector a= {1, 2, 3};  // yatay vektör
   vector b= {4, 5, 6};  // dikey vektör
   Print("a = ", a);
   Print("b = ", b);
   Print("1) a.MatMul(b) = ", a.MatMul(b));
   //--- Dot metodunun da aynı sonucu ürettiğini göster
   Print("2) a.Dot(b) = ", a.Dot(b));

  /* Sonuç
   a = [1,2,3]
   b = [4,5,6]
   1) a.MatMul(b) = 32.0
   2) a.Dot(b) = 32.0
    */
  }
```

Ayrıca bakınız

Dot, GeMM

GeMM

Genel matris çarpımı (General Matrix Multiply, GeMM) yöntemi, iki matrisin genel çarpımını uygular. İşlem C ← α A B + β C olarak tanımlanır, burada A ve B matrisleri isteğe bağlı olarak devrik olabilir. AB matrislerinin (MatMul) normal çarpımı ile *alpha* skalerin 1'e eşit olduğu, *beta*'nın da sıfıra eşit olduğu varsayılır.

Verimlilik açısından GeMM ve MatMul arasındaki temel fark, MatMul'un her zaman yeni bir matris/vektör nesnesi oluşturması, GeMM'in ise mevcut matris nesnesi üzerinde çalışması, onu yeniden oluşturmamasıdır. Bu nedenle, GeMM'i kullandığınızda ve ilgili matris için belleği önceden tahsis ettiğinizde, devamında aynı matris büyüklüğüyle çalışırken bellek yeniden tahsisi olmayacaktır. Bu, örneğin strateji sınayıcıda optimizasyon veya sinir ağı eğitimi gerçekleştirilmesi gibi toplu hesaplamalarda GeMM'in önemli bir avantajı olabilir.

MatMul'a benzer şekilde, GeMM de 4 aşırı yüke sahiptir. Ancak, dikey vektörün yatay vektörle çarpılmasının sağlanması amacıyla dördüncü aşırı yükün anlamı değiştirilmiştir.

Mevcut bir matris/vektör nesnesinde, belleği önceden tahsis etmek gerekli değildir. Bellek, ilk GeMM çağrısında tahsis edilecek ve 0'larla doldurulacaktır.

Bir matrisi bir matrisle çarpma: `matrix C[M][N] = α * ( matrix A[M][K] * matrix B[K][N]) + β * matrix C[M][N]`

```
bool  matrix::GeMM(
  const matrix &A,     // birinci matris
  const matrix &B,     // ikinci matris
  double alpha,        // AB çarpımı için alpha çarpanı
  double beta,         // C matrisi için beta çarpanı
  uint   flags         // A, B ve C matrislerinin devrik olup olmadığını tanımlayan ENU
   );
```

Bir vektörü bir matrisle çarpma: `vector C[N] = α * ( vector A[K] * matrix B[K][N]) + β * vector C[N]`

```
bool  vector::GeMM(
  const vector &A,     // yatay vektör
  const matrix &B,     // matris
  double alpha,        // AB çarpımı için alpha çarpanı
  double beta,         // C vektörü için beta çarpanı
  uint   flags         // A matrisinin devrik olup olmadığını tanımlayan ENUM_GEMM numa
   );
```

Bir matrisi bir vektörle çarpma: `vector C[M] = α * ( matrix A[M][K] * vector B[K] * ) + β * vector C[M]`

```
bool  vector::GeMM(
  const matrix &A,     // matris
  const vector &B,     // dikey vektör
  double alpha,        // AB çarpımı için alpha çarpanı
  double beta,         // C vektörü için beta çarpanı
  uint   flags         // B matrisinin devrik olup olmadığını tanımlayan ENUM_GEMM numa
   );
```

Bir vektörü bir vektörle çarpma: `matrix C[M][N] = α * ( vector A[M] * vector B[N] * ) + β * matrix C[M][N].` Bu aşırı yük, bir skalerin geri döndürüldüğü MatMul'un aksine, bir matris geri döndürür.

```
bool  matrix::GeMM(
  const vector &A,     // birinci vektör
  const vector &B,     // ikinci vektör
  double alpha,        // AB çarpımı için alpha çarpanı
  double beta,         // C matrisi için beta çarpanı
  uint   flags         // C matrisinin devrik olup olmadığını tanımlayan ENUM_GEMM numa
   );
```

Parametreler

A

[in] Matris veya vektör.

B

[in] Matris veya vektör.

alpha

[in] AB çarpımı için alpha çarpanı.

beta

[in] Ortaya çıkan C matrisi için beta çarpanı.

flags

[in] A, B ve C matrislerinin devrik olup olmadığını tanımlayan ENUM_GEMM numaralandırması değeri.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

ENUM_GEMM

GeMM metodu için bayrakların numaralandırması.

Kimlik	Açıklama
TRANSP_A	Devrik A matrisini kullan
TRANSP_B	Devrik B matrisini kullan
TRANSP_C	Devrik C matrisini kullan

Not

float, double ve complex türdeki matrisler ve vektörler, A ve B parametreleri olarak kullanılabilir. GeMM metodunun şablon varyantları aşağıdaki gibidir:

```
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)
```

Temel olarak genel matris çarpımı fonksiyonu şu şekilde tanımlanır:

```
C[m,n] = α *Sum(A[m,k]*B[k,n]) + β*C[m,n]
```

Burada A matrisi M x K, B matrisi K x N ve C matrisi de M x N büyüklüğündedir.

Bu nedenle, matrisler çarpma işlemi için uyumlu olmalıdır, yani birinci matristeki sütun sayısı, ikinci matristeki satır sayısına eşit olmalıdır. Matris çarpımı değişmeli değildir: genel olarak, birinci matrisin ikinci matrisle çarpımının sonucu, ikinci matrisin birinci matrisle çarpımının sonucuna eşit değildir.

Örnek:

```
void OnStart()
  {
   vector vector_a= {1, 2, 3, 4, 5};
   vector vector_b= {4, 3, 2, 1};
   matrix matrix_c;
//--- iki vektör için GeMM hesapla
   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]]
   */
//--- vektör biçiminde matrisler oluştur
   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]]
    */
//-- iki matris için GeMM hesapla ve aynı sonucu al
    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]]
    */
  }
```

Ayrıca bakınız

MatMul

Power

Kare matrisi bir tam sayı kuvvete yükseltir.

```
matrix matrix::Power(
  const int  power      // kuvvet
   );
```

Parametreler

power

[in] Üs herhangi bir tam sayı olabilir: pozitif, negatif veya sıfır.

Geri dönüş değeri

Matris.

Not

Ortaya çıkan matris, orijinal matris ile aynı büyüklüğe sahiptir. 0 kuvvetine yükseltildiğinde, birim matris geri döndürülür. n pozitif kuvveti, orijinal matrisin kendisiyle n kez çarpıldığı anlamına gelir. -n negatif kuvveti ise orijinal matrisin önce ters çevrildiği ve ardından bu ters matrisin kendisiyle n kez çarpıldığı anlamına gelir.

MQL5'te bir matrisi bir güce yükseltmek için basit bir algoritma:

```
bool MatrixPower(matrix& c, const matrix& a, const int power)
  {
//--- matris kare matris olmalıdır
   if(a.Rows()!=a.Cols())
      return(false);
//--- elde edilen matrisin büyüklüğü tam olarak aynıdır
   ulong  rows=a.Rows();
   ulong  cols=a.Cols();
   matrix result(rows,cols);
//--- sıfıra yükseltildiğinde, birim matris geri döndürülür
   if(power==0)
      result.Identity();
   else
     {
      //--- negatif üs için önce matrisi ters çevir
      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);
  }
```

MQL5 örneği:

```
  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]]
  */
```

Python örneği:

```
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

İki vektörün nokta çarpımı.

```
double vector::Dot(
  const vector&  b      // ikinci vektör
   );
```

Parametreler

b

[in] Vektör.

Geri dönüş değeri

Skaler.

Not

İki matrisin nokta çarpımı, matrix::MatMul() matris çarpımıdır.

MQL5'te vektörlerin skaler çarpımı için basit bir algoritma:

```
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);
  }
```

MQL5 örneği:

```
   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);
        }
     }
```

Python örneği:

```
import numpy as np

a = [1, 0, 0, 1]
b = [4, 1, 2, 2]
print(np.dot(a, b))

>>> 6
```

Kron

İki matrisin, matrisin ve vektörün, vektörün ve matrisin veya iki vektörün Kronecker çarpımını geri döndürür.

```
matrix matrix::Kron(
  const matrix&  b      // ikinci matris
   );

matrix matrix::Kron(
  const vector&  b      // vektör
   );

matrix vector::Kron(
  const matrix&  b      // matris
   );

matrix vector::Kron(
  const vector&  b      // ikinci vektör
   );
```

Parametreler

b

[in] İkinci matris.

Geri dönüş değeri

Matris.

Not

Kronecker çarpımı aynı zamanda blok matris çarpımı olarak da adlandırılır.

MQL5'te iki matrisin Kronecker çarpımını hesaplamak için basit bir algoritma:

```
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);
  }
```

MQL5 örneği:

```
   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]]
  */
```

Python örneği:

```
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

İki matrisin iç çarpımı.

```
matrix matrix::Inner(
  const matrix&  b      // ikinci matris
   );
```

Parametreler

b

[in] Matris.

Geri dönüş değeri

Matris.

Not

İki vektörün iç çarpımı, iki vektörün vector::Dot() nokta çarpımıdır.

MQL5'te iki matrisin iç çarpımını hesaplamak için basit bir algoritma:

```
bool MatrixInner(matrix& c, const matrix& a, const matrix& b)
  {
//--- sütun sayısı eşit olmalıdır
   if(a.Cols()!=b.Cols())
      return(false);
//--- ortaya çıkan matrisin büyüklüğü, matrislerin her birindeki vektörlerin sayısına
   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);
  }
```

MQL5 örneği:

```
   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]]
  */
```

Python örneği:

```
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

İki matrisin veya iki vektörün dış çarpımını hesaplar.

```
matrix matrix::Outer(
  const matrix&  b      // ikinci matris
   );

matrix vector::Outer(
  const vector&  b      // ikinci vektör
   );
```

Parametreler

b

[in] Matris.

Geri dönüş değeri

Matris.

Not

Kronecker çarpımı gibi dış çarpım da bir blok matris (ve vektör) çarpımıdır.

MQL5'te iki matrisin dış çarpımını hesaplamak için basit bir algoritma:

```
matrix MatrixOuter(const matrix& matrix_a, const matrix& matrix_b)
  {
//--- ortaya çıkan matrisin büyüklüğü, matrislerin büyüklüklerine bağlıdır
   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);
  }
```

MQL5 örneği:

```
   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]]
   */
```

Python örneği:

```
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

Pearson korelasyon katsayısını (lineer korelasyon katsayısı) hesaplar.

```
matrix matrix::CorrCoef(
  const bool    rowvar=true  // true ise gözlem vektörleri satırlarda, false ise sütur
);

scalar vector::CorrCoef(
  const vector&  b           // ikinci vektör
);
```

Geri dönüş değeri

Pearson momentler çarpımı korelasyon katsayıları.

Not

Korelasyon katsayısı [-1, 1] aralığındadır.

Kayan nokta yuvarlaması nedeniyle, ortaya çıkan dizi Hermisyen olmayabilir, köşegen elemanları 1 olmayabilir ve elemanlar abs(a)<=1 eşitsizliğini karşılamayabilir. Bu durumu iyileştirmek amacıyla, reel ve sanal kısımlar [-1, 1] aralığına kırpılır, ancak karmaşık durumlarda bu pek yardımcı olmaz.

MQL5'te iki vektörün korelasyon katsayısını hesaplamak için basit bir algoritma:

```
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;
//--- hesapla
   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;
     }
//--- kontrol et
   if(xv==0 || yv==0)
      return(0);
//--- sonucu geri döndür
   return(s/(MathSqrt(xv)*MathSqrt(yv)));
  }
```

MQL5 örneği:

```
   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]]
   */
```

Python örneği:

```
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

Kovaryans matrisini hesaplar.

```
matrix matrix::Cov(
  const bool    rowvar=true  // true ise gözlem vektörleri satırlarda, false ise sütur
);

matrix vector::Cov(
  const vector&  b           // ikinci vektör
);
```

Parametreler

b

[in] İkinci vektör.

Not

Kovaryans matrisini hesaplar.

MQL5'te iki vektörün kovaryans matrisini hesaplamak için basit bir algoritma:

```
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());
//--- kontrol et
   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);
//--- hesapla
   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 ve dikkate alınmaz)
   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];
        }
     }
//--- simetriyi zorla
   for(i=0; i<m-1; i++)
      for(j=i+1; j<m; j++)
         matrix_c[j][i]=matrix_c[i][j];
//---
   return(true);
  }
```

MQL5 örneği:

```
   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]]
  */
```

Python örneği:

```
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

İki vektörün çapraz korelasyonunu hesaplar.

```
vector vector::Correlate(
  const vector&           v,        // vektör
  ENUM_VECTOR_CONVOLVE    mode      // mod
   );
```

Parametreler

v

[in] İkinci vektör.

mode

[in] mode parametresi, lineer konvolüsyon hesaplama modunu (ENUM_VECTOR_CONVOLVE) tanımlar.

Geri dönüş değeri

İki vektörün çapraz korelasyonu.

Not

mode parametresi lineer konvolüsyon hesaplama modunu tanımlar.

MQL5'te iki vektörün korelasyon katsayısını hesaplamak için basit bir algoritma:

```
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);
  }
```

MQL5 örneği:

```
   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]
  */
```

Python örneği:

```
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

İki vektörün ayrık, lineer konvolüsyonunu geri döndürür

```
vector vector::Convolve(
  const vector&            v,         // vektör
  ENUM_VECTOR_CONVOLVE     mode       // mod
   );
```

Parametreler

v

[out] İkinci vektör.

mode

[in] mode parametresi, lineer konvolüsyon hesaplama modunu (ENUM_VECTOR_CONVOLVE) tanımlar.

Geri dönüş değeri

İki vektörün ayrık, lineer konvolüsyonu.

MQL5'te iki vektörün konvolüsyonunu hesaplamak için basit bir algoritma:

```
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);
  }
```

MQL5 örneği:

```
  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]
  */
```

Python örneği:

```
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]
```

Matris Dönüşümleri

Matris ayrışması aşağıdaki durumlarda kullanılabilir:

- Lineer denklem sistemlerini çözerken bir ara adım olarak.
- Matrisin tersini alırken.
- Determinantları hesaplarken.
- Matrisin özdeğerlerini ve özvektörlerini bulurken.
- Matrislerin analitik fonksiyonlarını hesaplarken.
- En küçük kareler yöntemini kullanırken.
- Diferansiyel denklemlerin sayısal çözümünde.

Probleme bağlı olarak farklı matris ayrışması türleri kullanılır.

Fonksiyon	Eylem
Cholesky	Cholesky ayrışmasını hesaplar.
Eig	Kare matrisin özdeğerlerini ve sağ özvektörlerini hesaplar
EigVals	Genel matrisin özdeğerlerini hesaplar
LU	Alt üçgen matris ve üst üçgen matrisin çarpımı olarak, matrisin LU ayrışması
LUP	Yalnızca satır permütasyonlarıyla LU ayrışması olarak, kısmi pivotlu LUP ayrışması: PA=LU
QR	Matrisin qr ayrışmasını hesaplar
SVD	Tekil değer ayrışması

Cholesky

Cholesky ayrışmasını hesaplar.

```
bool matrix::Cholesky(
  matrix&  L      // matris
   );
```

Parametreler

L

[out] Alt üçgen matris.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Not

a kare matrisinin Cholesky ayrışmasını, L * L.H, geri döndürür; burada L alt üçgen matristir, L.H ise bu matrisin eşlenik devriğidir (a reel değerliyse sıradan devriğidir). a Hermisyen (reel değerliyse simetrik) ve pozitif tanımlı olmalıdır. a'nın Hermisyen olup olmadığını doğrulamak için kontrol yapılmaz. Ayrıca, a'nın sadece alt üçgen ve köşegen elemanları kullanılır. Aslında sadece L geri döndürülür.

Örnek:

```
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

Kare matrisin özdeğerlerini ve sağ özvektörlerini hesaplar.

```
bool matrix::Eig(
  matrix&  eigen_vectors,     // özvektörler matrisi
  vector&  eigen_values       // özdeğerler vektörü
   );
```

Parametreler

eigen_vectors

[out] Dikey özvektörlerin matrisi.

eigen_values

[out] Özdeğerler vektörü.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Örnek:

```
#property script_show_inputs
//--- girdi parametreleri
input int  InpSize1   =512;
input int  InpSize2   =256;
input int  InpSize3   =1024;
//+------------------------------------------------------------------+
//| Tersinir kare bir test matrisini doldur                          |
//+------------------------------------------------------------------+
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;
//--- matrisi doldur
  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;
       }
     }
```

```
   }
 }
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
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);
 }

/*
Sonuç

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
*/

//+------------------------------------------------------------------+
//|  Eig metodunu test et                                            |
//+------------------------------------------------------------------+
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);
//--- kare matrisi doldur
  MatrixFill(matrix_a);
//--- mikrosaniye ölç
  ulong t1=GetMicrosecondCount();
//--- öz çözücü
  matrix_a.Eig(matrix_v, vector_e);
```

```
//--- ölç
  ulong t2=GetMicrosecondCount();
//--- A * v = lambda * v olup olmadığını kontrol et
  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);

    //--- çok fazla bölme, hassasiyet kontrolünü 10. basamağa zayıflat
    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_ti
  return(vectors==0);
 }
```

EigVals

Genel matrisin özdeğerlerini hesaplar.

```
bool matrix::EigVals(
  vector&  eigen_values      // özdeğerler vektörü
   );
```

Parametreler

eigen_values

[out] Sağ özdeğerler vektörü.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Not

EigVals ve Eig arasındaki tek fark, EigVals metodunun özvektörleri hesaplamaması, sadece özdeğerleri hesaplamasıdır.

LU

Alt üçgen matris ve üst üçgen matrisin çarpımı olarak, matrisin LU ayrışması.

```
bool matrix::LU(
  matrix&  L,     // alt üçgen matris
  matrix&  U      // üst üçgen matris
   );
```

Parametreler

L

[out] Alt üçgen matris.

U

[out] Üst üçgen matris.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Örnek:

```
   matrix matrix_a={{1,2,3,4},
                    {5,2,6,7},
                    {8,9,3,10},
                    {11,12,14,4}};
   matrix matrix_l,matrix_u;
//--- LU ayrışması
   matrix_a.LU(matrix_l,matrix_u);
   Print("matrix_l\n",matrix_l);
   Print("matrix_u\n",matrix_u);
//--- A = L * U olup olmadığını kontrol et
   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

Yalnızca satır permütasyonlarıyla LU ayrışması olarak, kısmi pivotlu LUP ayrışması: PA=LU.

```
bool  LUP(
  matrix&  L,      // alt üçgen matris
  matrix&  U,      // üst üçgen matris
  matrix&  P       // permütasyonlar matrisi
   );
```

Parametreler

L

[out] Alt üçgen matris.

U

[out] Üst üçgen matris.

P

[out] Permütasyonlar matrisi.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Örnek:

```
   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;
//--- LUP ayrışması
   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);
//--- P * A = L * U olup olmadığını kontrol et
   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

Matrisin qr ayrışmasını hesaplar.

```
bool  QR(
  matrix&  Q,      // ortonormal sütunlu matris
  matrix&  R       // üst üçgen matris
   );
```

Parametreler

Q

[out] Ortonormal sütunlu matris. mod = complete olduğunda sonuç, a'nın reel/karmaşık olup olmamasına bağlı olarak ortogonal/üniter bir matristir. Bu durumda determinant +/- 1 olabilir. Girdi dizisindeki boyut sayısı 2'den büyükse, yukarıdaki özelliklere sahip bir matris yığını geri döndürülür.

R

[out] Üst üçgen matris.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Örnek:

```
//---  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,-0.03745747856526496,0.7041241452319315,0.5374574785652647]
  [-0.5374574785652648,-0.03745747856526496,0.7041241452319316,-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

Tekil değer ayrışması.

```
bool matrix::SVD(
  matrix&  U,                    // üniter matris
  matrix&  V,                    // üniter matris
  vector&  singular_values       // tekil değerler vektörü
   );
```

Parametreler

U

[out] Sol tekil vektörlerden oluşan m mertebesinden üniter matris.

V

[out] Sağ tekil vektörlerden oluşan n mertebesinden üniter matris.

singular_values

[out] Tekil değerler.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Örnek:

```
  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);
//--- SVD ayrışması
  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);

// kontrol bloğu
//--- U * tekil köşegen * 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);

//---- başka bir kontrol
  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]
  */
 }
```

İstatistik Metotları

Matrislerin ve vektörlerin tanımlayıcı istatistiklerini hesaplama metotları.

Fonksiyon	Eylem
ArgMax	Maksimum değerin indeksini geri döndürür
ArgMin	Minimum değerin indeksini geri döndürür
Max	Matristeki/vektördeki maksimum değeri geri döndürür
Min	Matristeki/vektördeki minimum değeri geri döndürür
Ptp	Matrisin/vektörün veya belirtilen matris ekseninin değer aralığını geri döndürür
Sum	Belirtilen eksen(ler) boyunca da gerçekleştirilebilen matris/vektör elemanlarının toplamını geri döndürür
Prod	Belirtilen eksen boyunca da gerçekleştirilebilen matris/vektör elemanlarının çarpımını geri döndürür
CumSum	Belirtilen eksen boyunca olanlar da dahil olmak üzere matris/vektör elemanlarının kümülatif toplamını geri döndürür
CumProd	Belirtilen eksen boyunca olanlar da dahil olmak üzere matris/vektör elemanlarının kümülatif çarpımını geri döndürür
Percentile	Matris/vektör elemanlarının veya belirtilen eksen boyuncaki elemanların belirtilen yüzdelik dilimini geri döndürür
Quantile	Matris/vektör elemanlarının veya belirtilen eksen boyuncaki elemanların belirtilen dağılım dilimini geri döndürür
Median	Matris/vektör elemanlarının medyanını hesaplar
Mean	Elemanların aritmetik ortalamasını hesaplar
Average	Matris/vektör değerlerinin ağırlıklı ortalamasını hesaplar
Std	Matris/vektör elemanlarının veya belirtilen eksen boyuncaki elemanların standart sapmasını geri döndürür
Var	Matris/vektör elemanlarının varyansını hesaplar
LinearRegression	Hesaplanan lineer regresyon değerleriyle bir vektör/matris hesaplar

ArgMax

Maksimum değerin indeksini geri döndürür.

```
ulong vector::ArgMax();

ulong matrix::ArgMax();

vector matrix::ArgMax(
  const int  axis      // eksen
   );
```

Parametreler

axis

[in] Eksen. 0 - yatay eksen, 1 - dikey eksen.

Geri dönüş değeri

Maksimum değerin indeksi.

Örnek:

```
   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

Minimum değerin indeksini geri döndürür.

```
ulong vector::ArgMin();

ulong matrix::ArgMin();

vector matrix::ArgMin(
  const int  axis      // eksen
   );
```

Parametreler

axis

[in] Eksen. 0 - yatay eksen, 1 - dikey eksen.

Geri dönüş değeri

Minimum değerin indeksi.

Örnek:

```
   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

Matristeki/vektördeki maksimum değeri geri döndürür.

```
double vector::Max();

double matrix::Max();

vector matrix::Max(
  const int  axis      // eksen
   );
```

Parametreler

axis

[in] Eksen. 0 - yatay eksen, 1 - dikey eksen.

Geri dönüş değeri

Matristeki/vektördeki maksimum değer.

Örnek:

```
   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

Matristeki/vektördeki minimum değeri geri döndürür.

```
double vector::Min();

double matrix::Min();

vector matrix::Min(
  const int  axis      // eksen
   );
```

Parametreler

axis

[in] Eksen. 0 - yatay eksen, 1 - dikey eksen.

Geri dönüş değeri

Matristeki/vektördeki minimum değer.

Örnek:

```
   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

Matrisin/vektörün veya belirtilen matris ekseninin Max() - Min()'e eşit olan değer aralığını geri döndürür. Ptp: Peak to peak (tepeden tepeye).

```
double vector::Ptp();

double matrix::Ptp();

vector matrix::Ptp(
  const int  axis      // eksen
   );
```

Parametreler

axis

[in] Eksen. 0 - yatay eksen, 1 - dikey eksen.

Geri dönüş değeri

Değer aralığını içeren vektör (maksimum - minimum).

Örnek:

```
   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

Belirtilen eksen(ler) boyunca da gerçekleştirilebilen matris/vektör elemanlarının toplamını geri döndürür.

```
double vector::Sum();

double matrix::Sum();

vector matrix::Sum(
  const int  axis      // eksen
   );
```

Parametreler

axis

[in] Eksen. 0 - yatay eksen, 1 - dikey eksen.

Geri dönüş değeri

Belirtilen eksen(ler) boyunca da gerçekleştirilebilen matris/vektör elemanlarının toplamı.

Örnek:

```
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

Belirtilen eksen boyunca da gerçekleştirilebilen matris/vektör elemanlarının çarpımını geri döndürür.

```
double vector::Prod(
  const double  initial=1      // ilk çarpan
   );

double matrix::Prod(
  const double  initial=1      // ilk çarpan
   );

vector matrix::Prod(
  const int     axis,          // eksen
  const double  initial=1      // ilk çarpan
   );
```

Parametreler

axis

[in] Eksen. 0 - yatay eksen, 1 - dikey eksen.

initial=1

[in] İlk çarpan.

Örnek:

```
   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=[42,132,86.40000000000001]
   rows_prod=[60,96,120,693]
```

```
   prod value 479001600.0
*/
```

CumSum

Belirtilen eksen boyunca olanlar da dahil olmak üzere matris/vektör elemanlarının kümülatif toplamını geri döndürür.

```
vector vector::CumSum();

vector matrix::CumSum();

matrix matrix::CumSum(
  const int  axis      // eksen
   );
```

Parametreler

axis

[in] Eksen. 0 - yatay eksen, 1 - dikey eksen.

Geri dönüş değeri

Matris/vektör elemanlarının belirtilen eksen boyunca kümülatif toplamı.

Örnek:

```
   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

Belirtilen eksen boyunca olanlar da dahil olmak üzere matris/vektör elemanlarının kümülatif çarpımını geri döndürür.

```
vector vector::CumProd();

vector matrix::CumProd();

matrix matrix::CumProd(
  const int  axis      // eksen
   );
```

Parametreler

axis

[in] Eksen. 0 - her sütun için yatay eksen (yani satırların üzerinde), 1 - her satır için dikey eksen (yani sütunların üzerinde).

Geri dönüş değeri

Matris/vektör elemanlarının belirtilen eksen boyunca kümülatif çarpımı.

Örnek:

```
   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

Matris/vektör elemanlarının veya belirtilen eksen boyuncaki elemanların belirtilen yüzdelik dilimini geri döndürür.

```
double vector::Percentile(
  const int  percent      //
   );

double matrix::Percentile(
  const int  percent      //
   );

vector matrix::Percentile(
  const int  percent,     //
  const int  axis         // eksen
   );
```

Parametreler

percent

[in] Hesaplanacak yüzdelik dilimler, 0 ile 100 (dahil) arasında olmalıdır.

axis

[in] Eksen. 0 - yatay eksen, 1 - dikey eksen.

Geri dönüş değeri

Yüzdelik dilim: skaler veya vektör.

Not

percent parametresinin geçerli değerleri [0, 100] aralığındadır. Yüzdelik dilimleri hesaplamak için lineer bir algoritma kullanılır. Yüzdelik dilimlerin doğru bir şekilde hesaplanabilmesi adına elemanların sıralanması gerekir.

Örnek:

```
   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

Matris/vektör elemanlarının veya belirtilen eksen boyuncaki elemanların belirtilen dağılım dilimini geri döndürür.

```
double vector::Quantile(
  const double quantile      // dağılım dilimi
   );

double matrix::Quantile(
  const double quantile      // dağılım dilimi
   );

vector matrix::Quantile(
  const double quantile,     // dağılım dilimi
  const int    axis          // eksen
   );
```

Parametreler

quantile

[in] Hesaplanacak dağılım dilimleri, 0 ile 1 (dahil) arasında olmalıdır.

axis

[in] Eksen. 0 - yatay eksen, 1 - dikey eksen.

Geri dönüş değeri

Dağılım dilimi: skaler veya vektör.

Not

quantile parametresi [0, 1] aralığında değerler alır. Dağılım dilimlerini hesaplamak için lineer bir algoritma kullanılır. Dağılım dilimlerinin doğru bir şekilde hesaplanabilmesi adına elemanların sıralanması gerekir.

Örnek:

```
   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

Matris/vektör elemanlarının medyanını hesaplar.

```
double vector::Median();

double matrix::Median();

vector matrix::Median(
  const int  axis      // eksen
   );
```

Parametreler

axis

[in] Eksen. 0 - yatay eksen, 1 - dikey eksen.

Geri dönüş değeri

Medyan: skaler veya vektör.

Not

Medyan, matris/vektör elemanlarının yüksek yarısını düşük yarısından ayıran orta değerdir. Quantile(0,5) ve Percentile(50) ile aynıdır. Medyanın doğru bir şekilde hesaplanabilmesi adına elemanların sıralanması gerekir.

Örnek:

```
   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

Elemanların aritmetik ortalamasını hesaplar.

```
double vector::Mean();

double matrix::Mean();

vector matrix::Mean(
  const int  axis      // eksen
   );
```

Parametreler

axis

[in] Eksen. 0 - yatay eksen, 1 - dikey eksen.

Geri dönüş değeri

Elemanların aritmetik ortalaması.

Örnek:

```
   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

Matris/vektör değerlerinin ağırlıklı ortalamasını hesaplar.

```
double vector::Average(
  const vector& weigts      // ağırlıklar vektörü
   );

double matrix::Average(
  const matrix& weigts      // ağırlıklar matrisi
);

vector matrix::Average(
  const matrix& weigts,     // ağırlıklar matrisi
  const int     axis        // eksen
   );
```

Parametreler

axis

[in] Eksen. 0 - yatay eksen, 1 - dikey eksen.

Geri dönüş değeri

Ağırlıklı ortalama: skaler veya vektör.

Not

Ağırlıklar matrisi/vektörü, ana matris/vektör ile ilişkilendirilir.

Örnek:

```
   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

Matris/vektör elemanlarının veya belirtilen eksen boyuncaki elemanların standart sapmasını geri döndürür.

```
double vector::Std();

double matrix::Std();

vector matrix::Std(
  const int  axis      // eksen
   );
```

Parametreler

axis

[in] Eksen. 0 - yatay eksen, 1 - dikey eksen.

Geri dönüş değeri

Standart sapma: skaler veya vektör.

Not

Standart sapma, ortalamadan sapmaların karelerinin ortalamasının karekökü̈dür, yani std = sqrt(mean(x)), burada x = abs(a - a.mean())**2.

Sapmaların karelerinin ortalaması tipik olarak x.sum() / N olarak hesaplanır, burada N = len(x)'tir.

Örnek:

```
   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

Matris/vektör elemanlarının varyansını hesaplar.

```
double vector::Var();

double matrix::Var();

vector matrix::Var(
  const int  axis      // eksen
   );
```

Parametreler

axis

[in] Eksen. 0 - yatay eksen, 1 - dikey eksen.

Geri dönüş değeri

Varyans: skaler veya vektör.

Not

Varyans, ortalamadan sapmaların karelerinin ortalamasıdır, yani var = mean(x), burada x = abs(a - a.mean())**2.

Ortalama tipik olarak x.sum() / N olarak hesaplanır, burada N = len(x)'tir.

Örnek:

```
   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

Hesaplanan lineer regresyon değerleriyle bir vektör/matris hesaplar.

```
vector vector::LinearRegression();

matrix matrix::LinearRegression(
  ENUM_MATRIX_AXIS  axis=AXIS_NONE      // regresyonun hesaplandığı eksen
   );
```

Parametreler

axis

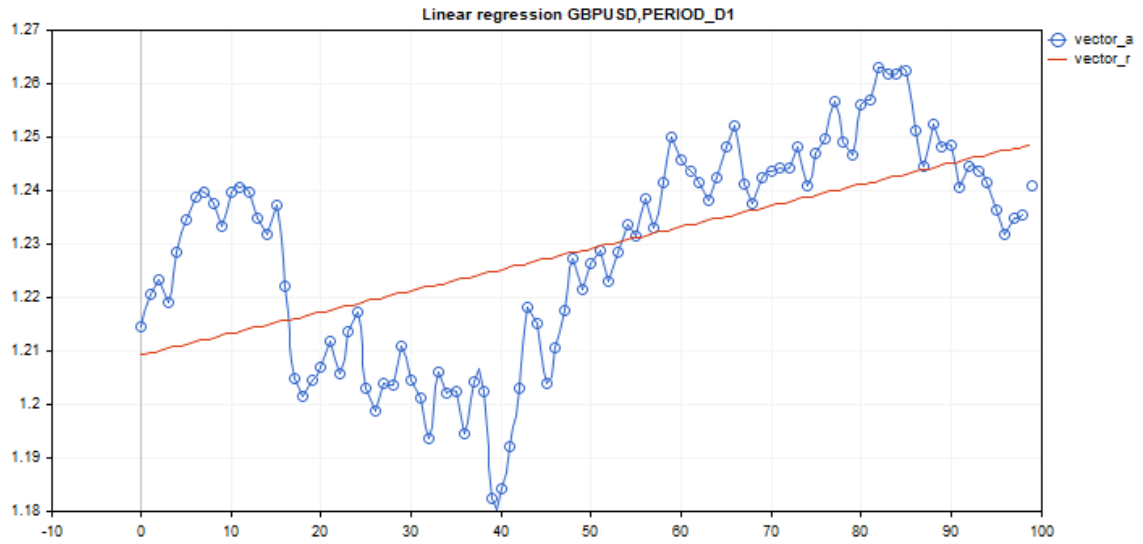
[in] Regresyonun hesaplandığı eksen. ENUM_MATRIX_AXIS numaralandırmasından değer (AXIS_HORZ - yatay eksen, AXIS_VERT - dikey eksen).

Geri dönüş değeri

Hesaplanan lineer regresyon değerlerine sahip vektör veya matris.

Not

Lineer regresyon, standart regresyon denklemi kullanılarak hesaplanır: y(x)=a*x+b, burada a çizgi eğimi, b ise Y ekseni kaymasıdır.



Örnek:

```
#include <Graphics\Graphic.mqh>

#define GRAPH_WIDTH  750
#define GRAPH_HEIGHT 350

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

```
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
void OnStart()
  {
   vector vector_a;
   vector_a.CopyRates(_Symbol,_Period,COPY_RATES_CLOSE,1,100);
   vector vector_r=vector_a.LinearRegression();

//--- grafik gösterimini kapat
   ChartSetInteger(0,CHART_SHOW,false);

//--- grafik çizmek için diziler
   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];
     }

//--- grafik başlığı
   string title="Linear regression "+_Symbol+","+EnumToString(_Period);

   long   chart=0;
   string name="LinearRegression";

//--- grafik oluştur
   CGraphic graphic;
   graphic.Create(chart,name,0,0,0,GRAPH_WIDTH,GRAPH_HEIGHT);
   graphic.BackgroundMain(title);
   graphic.BackgroundMainSize(12);

//--- aktivasyon fonksiyonu grafiği
   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);

//--- aktivasyon fonksiyonunun türevleri
   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();

//--- basılan klavye düğmelerini tanımak için sonsuz döngü
   while(!IsStopped())
     {
      //--- programdan çıkmak için escape düğmesine bas
      if(TerminalInfoInteger(TERMINAL_KEYSTATE_ESCAPE)!=0)
         break;
      //--- grafiğin görüntüsünü kaydetmek için PgDn düğmesine bas
      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);
     }

//--- temizle
   graphic.Destroy();
   ObjectDelete(chart,name);
   ChartSetInteger(0,CHART_SHOW,true);
  }
```

ENUM_MATRIX_AXIS

Matrisler için tüm istatistiksel fonksiyonlarda ekseni belirtmek için numaralandırma.

Kimlik	Açıklama
AXIS_NONE	Eksen belirtilmedi. Hesaplama, sanki bir vektörmüş gibi tüm matris elemanları üzerinden yapılır (Flat metoduna bakın)
AXIS_HORZ	Yatay eksen
AXIS_VERT	Dikey eksen

Özellik Metotları

Bu metotlar, matris özelliklerinin alınmasına olanak sağlar:

- Satır sayısı
- Sütun sayısı
- Norm
- Koşul sayısı
- Determinant
- Matris rankı
- İz
- Spektrum

Fonksiyon	Eylem
Rows	Matristeki satır sayısını geri döndürür
Cols	Matristeki sütun sayısını geri döndürür
Size	Vektörün büyüklüğünü geri döndürür
Norm	Matrisin veya vektörün normunu geri döndürür
Cond	Matrisin koşul sayısını hesaplar
Det	Tersinir kare matrisin determinantını hesaplar
SLogDet	Matrisin determinantının işaretini ve logaritmasını hesaplar
Rank	Gauss yöntemini kullanarak matris rankını geri döndürür
Trace	Matrisin köşegenlerinin toplamını geri döndürür
Spectrum	AT*A çarpımının özdeğerlerinin kümesi olarak, matrisin spektrumunu hesaplar

Rows

Matristeki satır sayısını geri döndürür.

```
ulong matrix::Rows()
```

Geri dönüş değeri

int türü değer.

Örnek:

```
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

Matristeki sütun sayısını geri döndürür.

```
ulong matrix::Cols()
```

Geri dönüş değeri

int türü değer.

Örnek:

```
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

Vektörün büyüklüğünü geri döndürür.

```
ulong vector::Size()
```

Geri dönüş değeri

int türü değer.

Örnek:

```
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

Matrisin veya vektörün normunu geri döndürür.

```
double vector::Norm(
  const ENUM_VECTOR_NORM  norm,     // vektör normu türü
  const int               norm_p=2  // VECTOR_NORM_P durumunda p sayısı
   );

double matrix::Norm(
  const ENUM_MATRIX_NORM  norm      // matris normu türü
   );
```

Parametreler

norm

[in] Norm türü.

Geri dönüş değeri

Matris veya vektör normu.

Not

- VECTOR_NORM_INF, vektör elemanları arasındaki maksimum mutlak değerdir.
- VECTOR_NORM_MINUS_INF, vektörün minimum mutlak değeridir.
- VECTOR_NORM_P, vektörün p-normudur. norm_p=0, sıfır olmayan vektör elemanlarının sayısıdır. norm_p=1, vektör elemanlarının mutlak değerlerinin toplamıdır. norm_p=2, vektör elemanlarının karelerinin toplamının karekköküdür. p sayısı negatif olabilir.
- MATRIX_NORM_FROBENIUS, matris elemanlarının karelerinin toplamının karekköküdür. Frobenius normu ve vektör 2-normu tutarlıdır.
- MATRIX_NORM_SPECTRAL, matris spektrumunun maksimum değeridir.
- MATRIX_NORM_NUCLEAR, matrisin tekil değerlerinin toplamıdır.
- MATRIX_NORM_INF, matrisin dikey vektörleri arasında maksimum vektör 1-normudur. Matris inf-normu ve vektör inf-normu tutarlıdır.
- MATRIX_NORM_MINUS_INF, matrisin dikey vektörleri arasında minimum vektör 1-normudur.
- MATRIX_NORM_P1, matrisin yatay vektörleri arasında maksimum vektör 1-normudur.
- MATRIX_NORM_MINUS_P1, matrisin yatay vektörleri arasında minimum vektör 1-normudur.
- MATRIX_NORM_P2, matrisin maksimum tekil değeridir.
- MATRIX_NORM_MINUS_P2, matrisin minimum tekil değeridir.

MQL5'te bir vektörün p-normunu hesaplamak için basit bir algoritma:

```
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);
  }
```

MQL5 örneği:

```
  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
  */
```

Python örneği:

```
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

Matrisin koşul sayısını hesaplar.

```
double matrix::Cond(
  const ENUM_MATRIX_NORM  norm      // matris normu türü
   );
```

Parametreler

norm

[in] ENUM_MATRIX_NORM numaralandırmasından matris normu türü.

Geri dönüş değeri

Matrisin koşul sayısı. Sonsuz olabilir.

Not

x'in koşul sayısı, x'in normu çarpı x'in tersinin normu olarak tanımlanır. Norm, olağan L2 normu (kareler toplamının karekökü) veya diğer matris normu türlerinden biri olabilir.

Koşul sayısı, A matrisinin normu ile A matrisinin tersinin normunun çarpımına eşit olan K değeridir. Koşul sayısı yüksek olan matrisler kötü koşullu olarak adlandırılır. Koşul sayısı düşük olanlar da iyi koşullu olarak ifade edilir. Ters matris, matrisin karesellik ve tekil olmama durumuyla sınırlandırılmaması için yalancı tersini alma işlemi yapılarak elde edilir.

Bir istisna, spektral koşul sayısıdır.

MQL5'te spektral koşul sayısını hesaplamak için basit bir algoritma:

```
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);
 }
```

MQL5 örneği:

```
 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
 */
```

Python örneği:

```
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

Tersinir kare matrisin determinantını hesaplar.

```
double matrix::Det()
```

Geri dönüş değeri

Matrisin determinantı.

Not

2. ve 3. mertebeden matris determinantları Sarrus kuralına göre hesaplanır. d2=a11*a22-a12*a21; d3=a11*a22*a33+a12*a23*a31+a13*a21*a32-a13*a22*a31-a11*a23*a32-a12*a21*a33.

Determinant, Gauss yöntemi kullanılarak matrisin üst üçgen formuna indirgenmesiyle hesaplanır. Üst üçgen matrisin determinantı, ana köşegen elemanların çarpımına eşittir.

Matrisin en az bir satırı veya sütunu sıfırsa, determinantı sıfırdır.

Matrisin iki veya daha fazla satırı veya sütunu lineer bağımlıysa, yine determinantı sıfırdır.

Matrisin determinantı, özdeğerlerinin çarpımına eşittir.

MQL5 örneği:

```
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
*/
```

Python örneği:

```
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

Matrisin determinantının işaretini ve logaritmasını hesaplar.

```
double matrix::SLogDet(
  int&  sign      // işaret
   );
```

Parametreler

sign

[out] Determinantın işareti. İşaret çift ise determinant pozitiftir.

Geri dönüş değeri

Determinantın işaretini temsil eden sayı.

Not

Determinant, Gauss yöntemi kullanılarak matrisin üst üçgen formuna indirgenmesiyle hesaplanır. Üst üçgen matrisin determinantı, ana köşegen elemanların çarpımına eşittir. Çarpımın logaritması, çarpılan elemanların logaritmalarının toplamına eşittir. Bu nedenle, determinant hesaplanırken aşırı eleman olması durumunda SLogDet metodunu kullanabilirsiniz.

İşaret çift ise determinant pozitiftir.

Örnek:

```
a = np.array([[1, 2], [3, 4]])
(sign, logdet) = np.linalg.slogdet(a)
(sign, logdet) (-1, 0.69314718055994529) # may vary  sign * np.exp(logdet) -2.0
```

Rank

Gauss yöntemini kullanarak matris rankını geri döndürür.

```
int  Rank()
```

Geri dönüş değeri

Matrisin rankı.

Not

m satır ve n sütuna sahip bir A matrisinin satır (veya sütun) sisteminin rankı, lineer bağımsız satırların (veya sütunların) maksimum sayısıdır. Diğerleri cinsinden lineer olarak ifade edilemiyorsa, bu satırlar (veya sütunlar) lineer bağımsız olarak adlandırılır. Satır sisteminin rankı her zaman sütun sisteminin rankına eşittir. Bu değere matrisin rankı denir.

MQL5 örneği:

```
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.;    // rank eksik matris
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 boyut - rank 1, tümü 0 olmadığı sürece

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
  */
```

Python örneği:

```
import numpy as np
from numpy.linalg import matrix_rank
a=(np.eye(4)) # tam rank matris
print("a \n", a)
print("matrix_rank(a)=",matrix_rank(a))
I=np.eye(4)
I[-1,-1] = 0. # rank eksik matris
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 boyut - rank 1, tümü 0 olmadığı sürece

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

Matrisin köşegenlerinin toplamını geri döndürür.

```
double matrix::Trace()
```

Geri dönüş değeri

Köşegen boyunca toplam.

Not

Bir matrisin izi, özdeğerlerinin toplamına eşittir.

MQL5 örneği:

```
  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
  */
```

Python örneği:

```
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

AT*A çarpımının özdeğerlerinin kümesi olarak, matrisin spektrumunu hesaplar.

```
vector matrix::Spectrum()
```

Geri dönüş değeri

Matris özdeğerlerinin vektörü olarak matrisin spektrumu.

Örnek:

```
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);
  }
```

Lineer Denklem Sistemlerini Çözmek için Matris Metotları

Lineer denklem sistemlerini çözme ve ters matrisi hesaplama metotları.

Fonksiyon	Eylem
Solve	Lineer matris denklemini veya lineer cebirsel denklem sistemini çözer
LstSq	Lineer cebirsel denklem sisteminin en küçük kareler çözümünü geri döndürür (kare olmayan veya dejenere matrisler için)
Inv	Jordan-Gauss yöntemiyle dejenere olmayan kare matrisin (çarpımsal) tersini hesaplar
PInv	Moore-Penrose yöntemiyle matrisin yalancı tersini hesaplar

Solve

Lineer matris denklemini veya lineer cebirsel denklem sistemini çözer.

```
vector matrix::Solve(
  const vector  b      // ordinat veya "bağımlı değişken" değerleri
   );
```

Parametreler

b

[in] Ordinat veya "bağımlı değişken" değerleri. (Serbest terimler vektörü).

Geri dönüş değeri

a * x = b sisteminin çözümüne sahip vektör.

Not

Matrisin en az bir satırı veya sütunu sıfırsa, sistemin çözümü yoktur.

Matrisin iki veya daha fazla satırı veya sütunu lineer bağımlıysa, yine sistemin çözümü yoktur.

Örnek:

```
//--- lineer cebirsel denklem sistemi (System of Linear Algebraic Equations, SLAE) çöz
   vector_x=matrix_a.Solve(vector_b);
//--- a * x = b olup olmadığını kontrol et
   result_vector=matrix_a.MatMul(vector_x);
   errors=vector_b.Compare(result_vector,1e-12);
```

LstSq

Lineer cebirsel denklem sisteminin en küçük kareler çözümünü geri döndürür (kare olmayan veya dejenere matrisler için).

```
vector matrix::LstSq(
  const vector  b      // ordinat veya "bağımlı değişken" değerleri
   );
```

Parametreler

b

[in] Ordinat veya "bağımlı değişken" değerleri. (Serbest terimler vektörü).

Geri dönüş değeri

a * x = b sisteminin çözümüne sahip vektör. Bu sadece kesin çözümü olan sistemler için geçerlidir.

Örnek:

```
   matrix a={{3, 2},
             {4,-5},
             {3, 3}};
   vector b={7,40,3};
//---
   vector x=a.LstSq(b);
//--- kontrol et, [5, -4] olmalıdır
   Print("x=", x);
//--- kontrol et, [7, 40, 3] olmalıdır
   vector b1=a.MatMul(x);
   Print("b1=",b1);

/*
  x=[5.000000000000002,-4]
  b1=[7.000000000000005,40.00000000000001,3.000000000000005]
*/
```

Inv

Jordan-Gauss yöntemiyle tersinir kare matrisin çarpımsal tersini hesaplar.

```
matrix matrix::Inv()
```

Geri dönüş değeri

Matrisin çarpımsal tersi.

Not

Orijinal matrisin ve ters matrisin çarpımı birim matristir.

Matrisin en az bir satırı veya sütunu sıfırsa, ters matris elde edilemez.

Matrisin iki veya daha fazla satırı veya sütunu lineer bağımlıysa, yine ters matris elde edilemez.

Örnek:

```
int TestInverse(const int size_m)
  {
   int    i,j,errors=0;
   matrix matrix_a(size_m,size_m);
//--- kare matrisi doldur
   MatrixTestFirst(matrix_a);
//--- mikrosaniye ölç
   ulong t1=GetMicrosecondCount();
//--- ters matrisi al
   matrix inverse=matrix_a.Inv();
//--- ölç
   ulong t2=GetMicrosecondCount();
//--- doğruluğu kontrol et
   matrix identity=matrix_a.MatMul(inverse);
//---
   for(i=0; i<size_m; i++)
     {
      for(j=0; j<size_m; j++)
        {
         double value;
         //--- köşegen boyunca 1'ler olmalıdır
         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);
                //--- çok fazla çarpma ve bölme, bu nedenle kontrol hassasiyetini azalt
                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,err
   return(errors);
  }
```

PInv

Moore-Penrose yöntemiyle matrisin yalancı tersini hesaplar.

```
matrix matrix::PInv()
```

Geri dönüş değeri

Matrisin yalancı tersi.

Örnek:

```
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;
//--- matrisi doldur
   MatrixTestFirst(matrix_a);
//--- tersini al
   matrix_inverted=matrix_a.PInv();
//--- doğruluğu kontrol et
   int errors=0;
//--- A * A+ * A = A (A+, A'nın yalancı tersidir)
   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":"fa
//---
   return(errors);
  }
```

Makine Öğrenimi

Bu metotlar makine öğreniminde kullanılır.

Sinir ağı aktivasyon fonksiyonu, girdilerin ağırlıklı toplamına bağlı olarak nöronun çıktı değerini belirler. Aktivasyon fonksiyonunun seçimi, sinir ağının performansı üzerinde büyük bir etkiye sahiptir. Modelin farklı bölümleri (katmanlar) farklı aktivasyon fonksiyonlarını kullanabilir.

Bilinen tüm aktivasyon fonksiyonlarına ek olarak, MQL5 onların türevlerini de sunar. Fonksiyon türevleri, öğrenme sırasında alınan hataya dayalı olarak model parametrelerinin verimli bir şekilde güncellenmesine olanak sağlar.

Bir sinir ağı, öğrenmedeki hatayı en aza indiren bir algoritma bulmayı amaçlar; işte burada kayıp fonksiyonu kullanılır. Kayıp fonksiyonunun değeri, model tarafından öngörülen değerin gerçek değerden ne kadar saptığını gösterir. Probleme bağlı olarak farklı kayıp fonksiyonları kullanılır. Örneğin, regresyon problemleri için "hataların karelerinin ortalaması (Mean Squared Error, MSE)" ve ikili sınıflandırma için "ikili çapraz entropi (Binary Cross Entropy, BCE)" kullanılır.

Fonksiyon	Eylem
Activation	Aktivasyon fonksiyonu değerlerini hesaplar ve onları aktarılan vektöre/matrise yazar
Derivative	Aktivasyon fonksiyonu türevinin değerlerini hesaplar ve onları aktarılan vektöre/matrise yazar
Loss	Kayıp fonksiyonu değerlerini hesaplar ve onları aktarılan vektöre/matrise yazar
LossGradient	Kayıp fonksiyonu gradyanlarının vektörünü veya matrisini hesaplar
RegressionMetric	Öngörülen verilerin kalitesinin doğru verilere kıyasla değerlendirilmesine olanak sağlayan regresyon metriğini hesaplar
ConfusionMatrix	Karışıklık matrisini hesaplar. Metot, öngörülen değerler vektörüne uygulanır
ConfusionMatrixMultilabel	Her etiket için karışıklık matrisini hesaplar. Metot, öngörülen değerler vektörüne uygulanır
ClassificationMetric	Öngörülen verilerin kalitesinin doğru verilere kıyasla değerlendirilmesine olanak sağlayan sınıflandırma metriğini hesaplar. Metot, tahmin edilen değerler vektörüne uygulanır
ClassificationScore	Öngörülen verilerin kalitesinin doğru verilere kıyasla değerlendirilmesine olanak sağlayan sınıflandırma metriğini hesaplar

Örnek:

Bu örnek, matris işlemlerini kullanarak bir modelin eğitimini göstermektedir. Model, (a + b + c)^2 / (a^2 + b^2 + c^2) fonksiyonu için eğitilmektedir. a, b ve c'nin farklı sütunlarda yer aldığı başlangıç veri matrisini girdi olarak giriyoruz. Modelin çıktısı, fonksiyonun sonucunu geri döndürecektir.

```
matrix weights1, weights2, weights3;              // ağırlıkların matrisleri
matrix output1, output2, result;                  // sinir katmanı çıktılarının matri
```

```
input int layer1 = 200;                                // ilk gizli katmanın büyüklüğü
input int layer2 = 200;                                // ikinci gizli katmanın büyüklüğü
input int Epochs = 20000;                              // eğitim dönemlerinin sayısı
input double lr = 3e-6;                                // öğrenme hızı
input ENUM_ACTIVATION_FUNCTION ac_func = AF_SWISH; // aktivasyon fonksiyonu
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int train = 1000;    // eğitim örneklemi büyüklüğü
   int test = 10;       // test örneklemi büyüklüğü
   matrix m_data, m_target;
//--- eğitim örneklemi oluştur
   if(!CreateData(m_data, m_target, train))
      return;
//--- modeli eğit
   if(!Train(m_data, m_target, Epochs))
      return;
//--- test örneklemi oluştur
   if(!CreateData(m_data, m_target, test))
      return;
//--- modeli test et
   Test(m_data, m_target);
  }
//+------------------------------------------------------------------+
//| Örneklem oluşturma metodu                                        |
//+------------------------------------------------------------------+
bool CreateData(matrix &data, matrix &target, const int count)
  {
//--- başlangıç veri ve sonuç matrislerini başlat
   if(!data.Init(count, 3) || !target.Init(count, 1))
      return false;
//--- başlangıç veri matrisini rastgele değerlerle doldur
   data.Random(-10, 10);
//--- eğitim örneklemi için hedef değerleri hesapla
   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;
//--- sonucu geri döndür
   return true;
  }
//+------------------------------------------------------------------+
//| Model eğitimi metodu                                             |
//+------------------------------------------------------------------+
bool Train(matrix &data, matrix &target, const int epochs = 10000)
  {
```

```
//--- modeli oluştur
   if(!CreateNet())
      return false;
//--- modeli eğit
   for(int ep = 0; ep < epochs; ep++)
     {
      //--- ileri besleme geçişi
      if(!FeedForward(data))
         return false;
      PrintFormat("Epoch %d, loss %.5f", ep, result.Loss(target, LOSS_MSE));
      //--- geri yayılım ve ağırlık matrisinin güncellenmesi
      if(!Backprop(data, target))
         return false;
     }
//--- sonucu geri döndür
   return true;
  }
//+------------------------------------------------------------------+
//| Model oluşturma metodu                                           |
//+------------------------------------------------------------------+
bool CreateNet()
  {
//--- ağırlık matrislerini başlat
   if(!weights1.Init(4, layer1) || !weights2.Init(layer1 + 1, layer2) || !weights3.Ini
      return false;
//--- ağırlık matrislerini rastgele değerlerle doldur
   weights1.Random(-0.1, 0.1);
   weights2.Random(-0.1, 0.1);
   weights3.Random(-0.1, 0.1);
//--- sonucu geri döndür
   return true;
  }
//+------------------------------------------------------------------+
//| İleri besleme metodu                                             |
//+------------------------------------------------------------------+
bool FeedForward(matrix &data)
  {
//--- başlangıç veri büyüklüğünü kontrol et
   if(data.Cols() != weights1.Rows() - 1)
      return false;
//--- ilk sinir katmanını hesapla
   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);
//--- aktivasyon fonksiyonunu hesapla
   if(!output1.Activation(temp, ac_func))
      return false;
```

```
//--- ikinci sinir katmanını hesapla
   if(!temp.Resize(temp.Rows(), weights2.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights2.Rows() - 1))
      return false;
   output2 = temp.MatMul(weights2);
//--- aktivasyon fonksiyonunu hesapla
   if(!output2.Activation(temp, ac_func))
      return false;
//--- üçüncü sinir katmanını hesapla
   if(!temp.Resize(temp.Rows(), weights3.Rows()) ||
      !temp.Col(vector::Ones(temp.Rows()), weights3.Rows() - 1))
      return false;
   result = temp.MatMul(weights3);
//--- sonucu geri döndür
   return true;
  }
//+------------------------------------------------------------------+
//| Geri yayılım metodu                                              |
//+------------------------------------------------------------------+
bool Backprop(matrix &data, matrix &target)
  {
//--- hedef değerler matrisinin büyüklüğünü kontrol et
   if(target.Rows() != result.Rows() ||
      target.Cols() != result.Cols())
      return false;
//--- hesaplanan değerlerin hedeften sapmasını belirle
   matrix loss = (target - result) * 2;
//--- gradyanı önceki katmana yay
   matrix gradient = loss.MatMul(weights3.Transpose());
//--- son katmanın ağırlık matrisini güncelle
   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;
//--- aktivasyon fonksiyonunun türeviyle hata gradyanını ayarla
   if(!output2.Derivative(temp, ac_func))
      return false;
   if(!gradient.Resize(gradient.Rows(), gradient.Cols() - 1))
      return false;
   loss = gradient * temp;
//--- gradyanı bir alt katmana yay
   gradient = loss.MatMul(weights2.Transpose());
//--- ikinci gizli katmanın ağırlık matrisini güncelle
   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;
//--- aktivasyon fonksiyonunun türeviyle hata gradyanını ayarla
   if(!output1.Derivative(temp, ac_func))
      return false;
   if(!gradient.Resize(gradient.Rows(), gradient.Cols() - 1))
      return false;
   loss = gradient * temp;
//--- ilk gizli katmanın ağırlık matrisini güncelle
   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;
//--- sonucu geri döndür
   return true;
  }
//+------------------------------------------------------------------+
//| Model test etme metodu                                           |
//+------------------------------------------------------------------+
bool Test(matrix &data, matrix &target)
  {
//--- test verileri üzerinde ileri besleme
   if(!FeedForward(data))
      return false;
//--- model hesaplama sonuçlarını ve doğru değerleri yazdır
   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]);
//--- sonucu geri döndür
   return true;
  }
//+------------------------------------------------------------------+
```

Activation

Aktivasyon fonksiyonu değerlerini hesaplar ve onları aktarılan vektöre/matrise yazar.

```
bool vector::Activation(
  vector&                   vect_out,      // değerlerin yazılacağı vektör
  ENUM_ACTIVATION_FUNCTION  activation,    // aktivasyon fonksiyonu türü
   ...                                     // ek parametreler
   );


bool matrix::Activation(
  matrix&                   matrix_out,    // değerlerin yazılacağı matris
  ENUM_ACTIVATION_FUNCTION  activation     // aktivasyon fonksiyonu türü
   );


bool matrix::Activation(
  matrix&                   matrix_out,    // değerlerin yazılacağı matris
  ENUM_ACTIVATION_FUNCTION  activation,    // aktivasyon fonksiyonu türü
  ENUM_MATRIX_AXIS          axis,          // eksen
   ...                                     // ek parametreler
   );
```

Parametreler

vect_out/matrix_out

[out] Aktivasyon fonksiyonunun hesaplanan değerlerinin yazılacağı vektör veya matris.

activation

[in] ENUM_ACTIVATION_FUNCTION numaralandırmasından aktivasyon fonksiyonu türü.

axis

[in] ENUM_MATRIX_AXIS numaralandırmasından değer (AXIS_HORZ - yatay eksen, AXIS_VERT - dikey eksen).

...

[in] Bazı aktivasyon fonksiyonları için gereken ek parametreler. Hiçbir parametre belirtilmezse, varsayılan değerler kullanılır.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Ek parametreler

Bazı aktivasyon fonksiyonları ek parametreler kabul eder. Hiçbir parametre belirtilmezse, varsayılan değerler kullanılır.

```
AF_ELU  (Exponential Linear Unit - Üssel lineer birim)
  double alpha=1.0

Aktivasyon fonksiyonu: if(x>=0) f(x) = x
                   else f(x) = alpha * (exp(x)-1)


AF_LINEAR
  double alpha=1.0
  double beta=0.0

Aktivasyon fonksiyonu: f(x) = alpha*x + beta


AF_LRELU   (Leaky REctified Linear Unit - Sızıntılı düzeltilmiş lineer birim)
  double alpha=0.3

Aktivasyon fonksiyonu: if(x>=0) f(x)=x
                   else f(x) = alpha*x


AF_RELU  (REctified Linear Unit - Düzeltilmiş lineer birim)
  double alpha=0.0
  double max_value=0.0
  double treshold=0.0

Aktivasyon fonksiyonu: 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

Aktivasyon fonksiyonu: f(x) = x / (1+exp(-x*beta))


AF_TRELU   (Thresholded REctified Linear Unit - Eşikli düzeltilmiş lineer birim)
  double theta=1.0

Aktivasyon fonksiyonu: if(x>theta) f(x) = x
                   else f(x) = 0


AF_PRELU   (Parametric REctified Linear Unit - Parametrik düzeltilmiş lineer birim)
  double alpha[] - learned array of coeefficients
```

```
Aktivasyon fonksiyonu: if(x[i]>=0) f(x)[i] = x[i]
                       else f(x)[i] = alpha[i] * x[i]
```

Not

Yapay sinir ağlarında, nöronun aktivasyon fonksiyonu, bir girdi sinyali veya bir girdi sinyali kümesi tarafından tanımlanan çıktı sinyalini belirler. Aktivasyon fonksiyonunun seçimi, sinir ağının performansı üzerinde büyük bir etkiye sahiptir. Modelin farklı bölümleri (katmanlar) farklı aktivasyon fonksiyonlarını kullanabilir.

Ek parametrelerin kullanımına ilişkin örnekler:

```
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);

/*  Sonuçlar
3   [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

Aktivasyon fonksiyonu türevinin değerlerini hesaplar ve onları aktarılan vektöre/matrise yazar

```
bool vector::Derivative(
  vector&                   vect_out,      // değerlerin yazılacağı vektör
  ENUM_ACTIVATION_FUNCTION  activation,    // aktivasyon fonksiyonu türü
   ...                                     // ek parametreler
   );


bool matrix::Derivative(
  matrix&                   matrix_out,    // değerlerin yazılacağı matris
  ENUM_ACTIVATION_FUNCTION  activation,    // aktivasyon fonksiyonu türü
   );


bool matrix::Derivative(
  matrix&                   matrix_out,    // değerlerin yazılacağı matris
  ENUM_ACTIVATION_FUNCTION  activation,    // aktivasyon fonksiyonu türü
  ENUM_MATRIX_AXIS          axis,          // eksen
   ...                                     // ek parametreler
   );
```

Parametreler

vect_out/matrix_out

[out] Aktivasyon fonksiyonu türevinin hesaplanan değerlerinin yazılacağı vektör veya matris.

activation

[in] ENUM_ACTIVATION_FUNCTION numaralandırmasından aktivasyon fonksiyonu türü.

axis

[in] ENUM_MATRIX_AXIS numaralandırmasından değer (AXIS_HORZ - yatay eksen, AXIS_VERT - dikey eksen).

...

[in] Buradaki ek parametreler, aktivasyon fonksiyonlarının ek parametreleriyle aynıdır. Yalnızca bazı aktivasyon fonksiyonları ek parametreler kabul eder. Hiçbir parametre belirtilmezse, varsayılan değerler kullanılır.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür.

Not

Fonksiyon türevleri, hata geri yayılımı sürecinde öğrenme sırasında alınan hataya dayalı olarak model parametrelerinin verimli bir şekilde güncellenmesine olanak sağlar.

Loss

Kayıp fonksiyonunun değerini hesaplar.

```
double vector::Loss(
  const vector&        vect_true,     // doğru değerlerin vektörü
  ENUM_LOSS_FUNCTION   loss,          // kayıp fonksiyonu türü
   ...                                // ek parametre
   );


double matrix::Loss(
  const matrix&        matrix_true,   // doğru değerlerin matrisi
  ENUM_LOSS_FUNCTION   loss,          // kayıp fonksiyonu türü
   );


double matrix::Loss(
  const matrix&        matrix_true,   // doğru değerlerin matrisi
  ENUM_LOSS_FUNCTION   loss,          // kayıp fonksiyonu türü
  ENUM_MATRIX_AXIS     axis,          // eksen
   ...                                // ek parametre
   );
```

Parametreler

vect_true/matrix_true

[in] Doğru değerlerin vektörü veya matrisi.

loss

[in] ENUM_LOSS_FUNCTION numaralandırmasından kayıp fonksiyonu türü.

axis

[in] ENUM_MATRIX_AXIS numaralandırmasından değer (AXIS_HORZ - yatay eksen, AXIS_VERT - dikey eksen).

...

[in] Yalnızca 'delta' ek parametresi Hubert kayıp fonksiyonu (LOSS_HUBER) tarafından kullanılabilir.

Geri dönüş değeri

double türü değer.

Hubert kayıp fonksiyonunda (LOSS_HUBER) 'delta' parametresi nasıl kullanılır?

```
   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);
```

Not

Bir sinir ağı, öğrenmedeki hatayı en aza indiren bir algoritma bulmayı amaçlar; işte burada kayıp fonksiyonu kullanılır.

Kayıp fonksiyonunun değeri, model tarafından öngörülen değerin gerçek değerden ne kadar saptığını gösterir.

Probleme bağlı olarak farklı kayıp fonksiyonları kullanılır. Örneğin, regresyon problemleri için "hataların karelerinin ortalaması (Mean Squared Error, MSE)" ve ikili sınıflandırma için "ikili çapraz entropi (Binary Cross Entropy, BCE)" kullanılır.

Hubert kayıp fonksiyonunu çağırma örneği:

```
   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);

/* Sonuç
   0.155
   0.15125
*/
```

LossGradient

Kayıp fonksiyonu gradyanlarının vektörünü veya matrisini hesaplar.

```
vector vector::LossGradient(
  const vector&       vect_true,     // doğru değerlerin vektörü
  ENUM_LOSS_FUNCTION  loss,          // kayıp fonksiyonu türü
   ...                               // ek parametre
   );


matrix matrix::LossGradient(
  const matrix&       matrix_true,   // doğru değerlerin matrisi
  ENUM_LOSS_FUNCTION  loss,          // kayıp fonksiyonu türü
   );


matrix matrix::LossGradient(
  const matrix&       matrix_true,   // doğru değerlerin matrisi
  ENUM_LOSS_FUNCTION  loss,          // kayıp fonksiyonu türü
  ENUM_MATRIX_AXIS    axis,          // eksen
   ...                               // ek parametre
   );
```

Parametreler

vect_true/matrix_true

[in] Doğru değerlerin vektörü veya matrisi.

loss

[in] ENUM_LOSS_FUNCTION numaralandırmasından kayıp fonksiyonu türü.

axis

[in] ENUM_MATRIX_AXIS numaralandırmasından değer (AXIS_HORZ - yatay eksen, AXIS_VERT - dikey eksen).

...

[in] Yalnızca 'delta' ek parametresi Hubert kayıp fonksiyonu (LOSS_HUBER) tarafından kullanılabilir.

Geri dönüş değeri

Kayıp fonksiyonu gradyan değerlerinin vektörü veya matrisi. Gradyan, belirli bir noktada kayıp fonksiyonunun dx'e (x öngörülen değerdir) göre kısmi türevidir.

Not

Gradyanlar, modeli eğitirken geri yayılım sırasında ağırlık matrisi ağırlıklarını ayarlamak için sinir ağlarında kullanılır.

Bir sinir ağı, öğrenmedeki hatayı en aza indiren bir algoritma bulmayı amaçlar; işte burada kayıp fonksiyonu kullanılır.

Probleme bağlı olarak farklı kayıp fonksiyonları kullanılır. Örneğin, regresyon problemleri için "hataların karelerinin ortalaması (Mean Squared Error, MSE)" ve ikili sınıflandırma için "ikili çapraz entropi (Binary Cross Entropy, BCE)" kullanılır.

Kayıp fonksiyonu gradyanlarını hesaplama örneği:

```
   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);

/* Sonuç
   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

Öngörülen verilerin kalitesinin doğru verilere kıyasla değerlendirilmesine olanak sağlayan regresyon metriğini hesaplar.

```
double vector::RegressionMetric(
   const vector&           vector_true,   // doğru değerlerin vektörü
   ENUM_REGRESSION_METRIC  metric         // metrik türü
   );

double matrix::RegressionMetric(
   const matrix&           matrix_true,   // doğru değerlerin matrisi
   ENUM_REGRESSION_METRIC  metric         // metrik türü
);

vector matrix::RegressionMetric(
   const matrix&           matrix_true,   // doğru değerlerin matrisi
   ENUM_REGRESSION_METRIC  metric,        // metrik türü
   int                     axis           // eksen
   );
```

Parametreler

vector_true/matrix_true

[in] Doğru değerlerin vektörü veya matrisi.

metric

[in] ENUM_REGRESSION_METRIC numaralandırmasından metrik türü.

axis

[in] Eksen. 0 - yatay eksen, 1 - dikey eksen.

Geri dönüş değeri

Öngörülen verilerin kalitesinin doğru verilere kıyasla değerlendirilmesine olanak sağlayan metrik.

Not

- REGRESSION_MAE - öngörülen değerler ile karşılık gelen doğru değerler arasındaki farkları temsil eden hataların mutlak değerlerinin ortalaması.
- REGRESSION_MSE - öngörülen değerler ile karşılık gelen doğru değerler arasındaki farkları temsil eden hataların karelerinin ortalaması.
- REGRESSION_RMSE - hataların karelerinin ortalamasının karekökü.
- REGRESSION_R2 - 1 - MSE(regresyon) / MSE(ortalama).
- REGRESSION_MAPE - yüzde olarak MAE.
- REGRESSION_MSPE - yüzde olarak MSE.
- REGRESSION_RMSLE - logaritmik ölçekte hesaplanan RMSE.

Örnek:

```
   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);

  /* Sonuç
   mae=0.375
   mse=0.5
   r2=0.9486081370449679
   */
```

ConfusionMatrix

Karışıklık matrisini hesaplar. Metot, öngörülen değerler vektörüne uygulanır.

```
matrix vector::ConfusionMatrix(
   const vector&        vect_true      // doğru değerler vektörü
   );


matrix vector::ConfusionMatrix(
   const vector&        vect_true,     // doğru değerler vektörü
   uint                 label          // etiket değeri
   );
```

Parametreler

vect_true

[in] Doğru değerler vektörü.

label

[in] Karışıklık matrisini hesaplamak için etiket değeri.

Geri dönüş değeri

Karışıklık matrisi. Bir etiket değeri belirtilmezse, her bir etiketin diğer her bir etiketle ayrı ayrı eşleştirildiği çok sınıflı bir karışıklık matrisi geri döndürülür. Bir etiket değeri belirtilirse, belirtilen etiketin pozitif, diğer tüm etiketlerin negatif (ovr, one vs rest) olarak kabul edildiği 2 x 2'lik bir matris geri döndürülür.

Not

Karışıklık matrisi C; Cij, i grubunda olduğu bilinen ve j grubunda olduğu öngörülen gözlemlerin sayısına eşit olacak şekildedir. Dolayısıyla, ikili sınıflandırmada doğru negatiflerin (TN) sayısı C00, yanlış negatiflerin (FN) sayısı C10, doğru pozitiflerin (TP) sayısı C11 ve yanlış pozitiflerin (FP) sayısı C01'dir.

Başka bir deyişle, matris grafiksel olarak aşağıdaki gibi gösterilebilir:

TN	FP
FN	TP

Doğru değerler vektörü ile öngörülen değerler vektörünün büyüklükleri aynı olmalıdır.

Örnek:

```
   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

Her etiket için karışıklık matrisini hesaplar. Metot, öngörülen değerler vektörüne uygulanır.

```
uint vector::ConfusionMatrixMultiLabel(
   const vector&       vect_true,     // doğru değerler vektörü
   matrix&             confusions[]   // hesaplanan karışıklık matrisleri dizisi
   );
```

Parametreler

vect_true

[in] Doğru değerler vektörü.

confusions

[out] Her etiket için hesaplanmış karışıklık matrislerini içeren 2 x 2'lik matrisler dizisi.

Geri dönüş değeri

Hesaplanan karışıklık matrisleri dizisinin büyüklüğü. Başarısızlık durumunda, 0 geri döner.

Not

Sonuç dizisi dinamik veya statik olabilir. Eğer dizi statik ise, sınıf sayısından daha az büyüklükte olmamalıdır.

Doğru değerler vektörü ile öngörülen değerler vektörünün büyüklükleri aynı olmalıdır.

Örnek:

```
   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

Öngörülen verilerin kalitesinin doğru verilere kıyasla değerlendirilmesine olanak sağlayan sınıflandırma metriğini hesaplar. Metot, tahmin edilen değerler vektörüne uygulanır.

```
vector vector::ClassificationMetric(
   const vector&               vect_true,     // doğru değerler vektörü
   ENUM_CLASSIFICATION_METRIC metric         // metrik türü
   );


vector vector::ClassificationMetric(
   const vector&               vect_true,     // doğru değerler vektörü
   ENUM_CLASSIFICATION_METRIC metric         // metrik türü
   ENUM_AVERAGE_MODE           mode           // ortalama alma modu
   );
```

Parametreler

vect_true

[in] Doğru değerler vektörü.

metric

[in] ENUM_CLASSIFICATION_METRIC numaralandırmasından metrik türü. CLASSIFICATION_TOP_K_ACCURACY, CLASSIFICATION_AVERAGE_PRECISION ve CLASSIFICATION_ROC_AUC dışındaki değerler uygulanır (ifade edilen değerler ClassificationScore metodunda kullanılır).

mode

[in] ENUM_AVERAGE_MODE numaralandırmasından ortalama alma modu. CLASSIFICATION_F1, CLASSIFICATION_JACCARD, CLASSIFICATION_PRECISION ve CLASSIFICATION_RECALL metrikleri için kullanılır.

Geri dönüş değeri

Hesaplanan metriği içeren vektör. AVERAGE_NONE ortalama alma modu durumunda, vektör ortalama alma olmadan her sınıf için metrik değerlerini içerir. (Örneğin, ikili sınıflandırma durumunda, sırasıyla 'false' ve 'true' için iki metrik olacaktır).

Ortalama alma modları hakkında not

AVERAGE_BINARY yalnızca ikili sınıflandırma için anlamlıdır.

AVERAGE_MICRO - toplam doğru pozitifleri, yanlış negatifleri ve yanlış pozitifleri sayarak metrikleri global olarak hesaplar.

AVERAGE_MACRO - her etiket için metrikleri hesaplar ve ağırlıklandırılmamış ortalamalarını bulur. Bu, etiket dengesizliğini hesaba katmaz.

AVERAGE_WEIGHTED - her etiket için metrikleri hesaplar ve destek (her etiket için doğru örnek sayısı) ile ağırlıklandırılmış ortalamalarını bulur. Bu, etiket dengesizliğini hesaba katmak için 'macro'yu değiştirir; kesinlik ve duyarlılık arasında olmayan bir F puanı ile sonuçlanabilir.

Not

İkili sınıflandırma durumunda, yalnızca 'ilk sütunun negatif bir etiket için olasılıkları ve ikinci sütunun pozitif bir etiket için olasılıkları içerdiği' n x 2'lik bir matrisi değil, aynı zamanda 'pozitif olasılıkları içeren bir sütundan oluşan' bir matrisi de girebiliriz. Bunun nedeni, ikili sınıflandırma modellerinin pozitif bir etiket için ya iki olasılık ya da bir olasılık geri döndürebilmesidir.

Örnek:

```
   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("");

//--- ikili sınıflandırma
   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

Öngörülen verilerin kalitesinin doğru verilere kıyasla değerlendirilmesine olanak sağlayan sınıflandırma metriğini hesaplar.

Makine öğrenimi bölümündeki diğer metotlardan farklı olarak, bu metot öngörülen değerler vektörü yerine doğru değerler vektörüne uygulanır.

```
vector vector::ClassificationScore(
   const matrix&               pred_scores,   // her sınıf için olasılık dağılımını içe
   ENUM_CLASSIFICATION_METRIC metric         // metrik türü
   ENUM_AVERAGE_MODE           mode          // ortalama alma modu
   );


vector vector::ClassificationScore(
   const matrix&               pred_scores,   // her sınıf için olasılık dağılımını içe
   ENUM_CLASSIFICATION_METRIC metric         // metrik türü
   int                         param         // ek parametre
   );
```

Parametreler

pred_scores

[in] Her sınıf için olasılıkları içeren bir dizi yatay vektöre sahip bir matris. Matris satırlarının sayısı, doğru değerler vektörünün büyüklüğüne karşılık gelmelidir.

metric

[in] ENUM_CLASSIFICATION_METRIC numaralandırmasından metrik türü. CLASSIFICATION_TOP_K_ACCURACY, CLASSIFICATION_AVERAGE_PRECISION ve CLASSIFICATION_ROC_AUC değerleri kullanılır.

mode

[in] ENUM_AVERAGE_MODE numaralandırmasından ortalama alma modu. CLASSIFICATION_AVERAGE_PRECISION ve CLASSIFICATION_ROC_AUC metrikleri için kullanılır.

param

[in] CLASSIFICATION_TOP_K_ACCURACY metriği söz konusu olduğunda, ortalama alma modu yerine tamsayı K değeri belirtilmelidir.

Geri dönüş değeri

Hesaplanan metriği içeren vektör. AVERAGE_NONE ortalama alma modu durumunda, vektör ortalama alma olmadan her sınıf için metrik değerlerini içerir. (Örneğin, ikili sınıflandırma durumunda, sırasıyla 'false' ve 'true' için iki metrik olacaktır).

Ortalama alma modları hakkında not

AVERAGE_BINARY yalnızca ikili sınıflandırma için anlamlıdır.

AVERAGE_MICRO - etiket gösterge matrisinin her bir elemanını bir etiket olarak değerlendirerek metrikleri global olarak hesaplar. Etiket gösterge matrisi, her etiket için bir dizi olasılık içeren bir matrisi ifade eder.

AVERAGE_MACRO - her etiket için metrikleri hesaplar ve ağırlıklandırılmamış ortalamalarını bulur. Bu, etiket dengesizliğini hesaba katmaz.

AVERAGE_WEIGHTED - her etiket için metrikleri hesaplar ve destek (her etiket için doğru örnek sayısı) ile ağırlıklandırılmış ortalamalarını bulur.

Not

İkili sınıflandırma durumunda, yalnızca 'ilk sütunun negatif bir etiket için olasılıkları ve ikinci sütunun pozitif bir etiket için olasılıkları içerdiği' n x 2'lik bir matrisi değil, aynı zamanda 'pozitif olasılıkları içeren bir sütundan oluşan' bir matrisi de girebiliriz. Bunun nedeni, ikili sınıflandırma modellerinin pozitif bir etiket için ya iki olasılık ya da bir olasılık geri döndürebilmesidir.

Örnek:

```
   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};

//--- etiket skorları 0 1 2 3 4 5 6 7 8 9 doğru öngörü
   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 ilk 2'de
                    {0.3, 0.4, 0.2},  // 1 ilk 2'de
                    {0.2, 0.4, 0.3},  // 2 ilk 2'de
                    {0.7, 0.2, 0.1}}; // 2 ilk 2'de değil
   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("");

//--- ikili sınıflandırma
   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_PRECI
   Print("average precision score binary = ",ap2);
   vector ap3=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_AVERAGE_PRECI
   Print("average precision score none = ",ap3);
   Print("");

   vector area=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_ROC_AUC,AVER
   Print("roc auc score binary = ",area);
   vector area2=y_true_bin.ClassificationScore(y_score1_bin,CLASSIFICATION_ROC_AUC,AVE
   Print("roc auc score binary = ",area2);
   vector area3=y_true_bin.ClassificationScore(y_scores_bin,CLASSIFICATION_ROC_AUC,AVE
   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);
     }    }
```

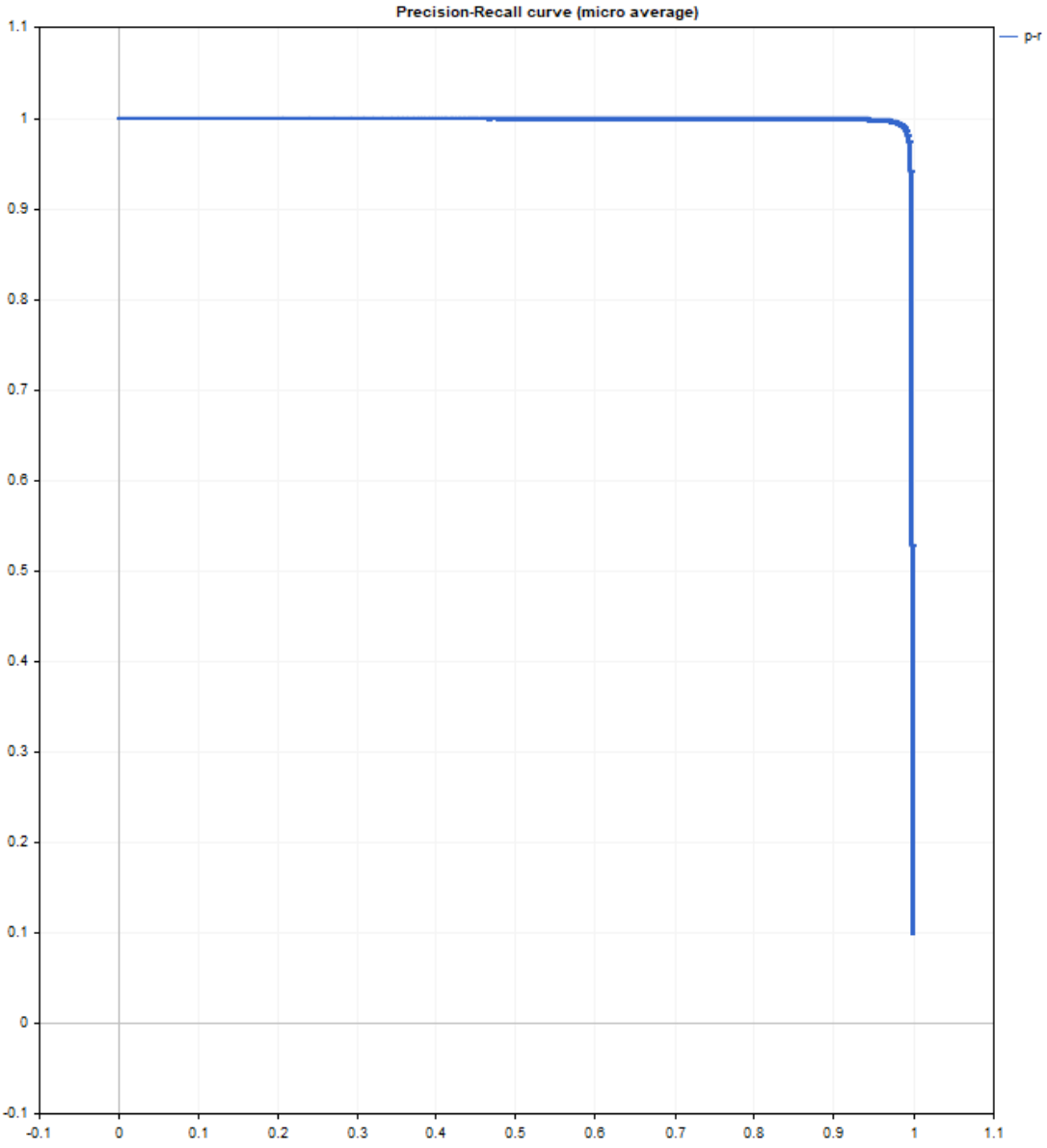
Accuracy calculation

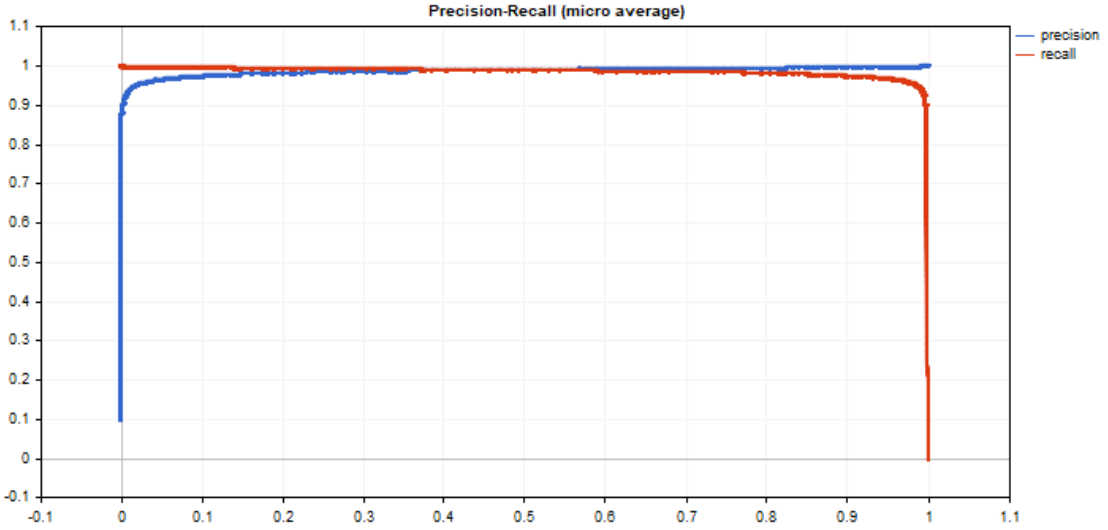
```
   vectorf accuracy=y_pred.ClassificationMetric(y_true,CLASSIFICATION_ACCURACY);
   PrintFormat("accuracy=%f",accuracy[0]);

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
   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.

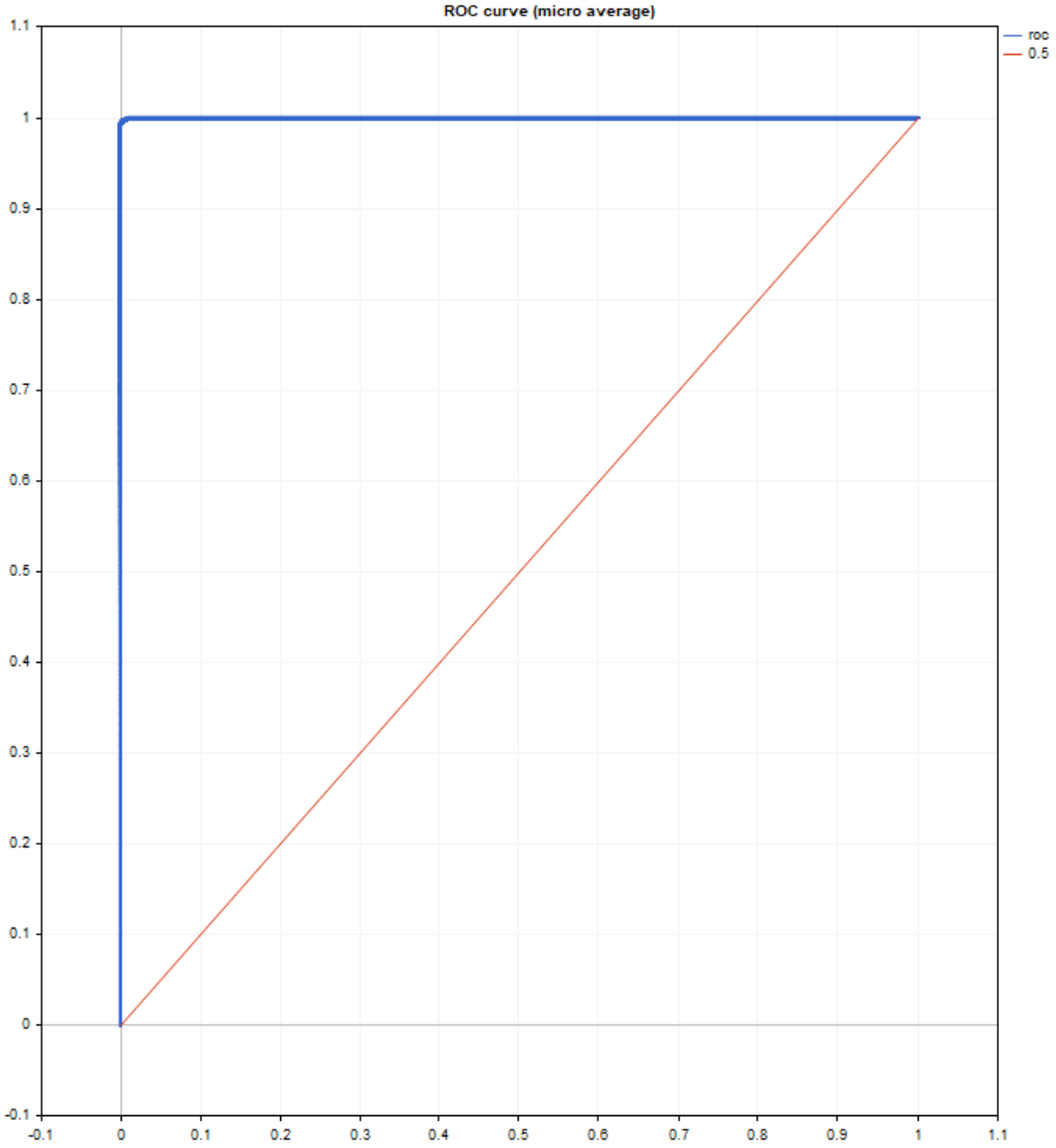
Example

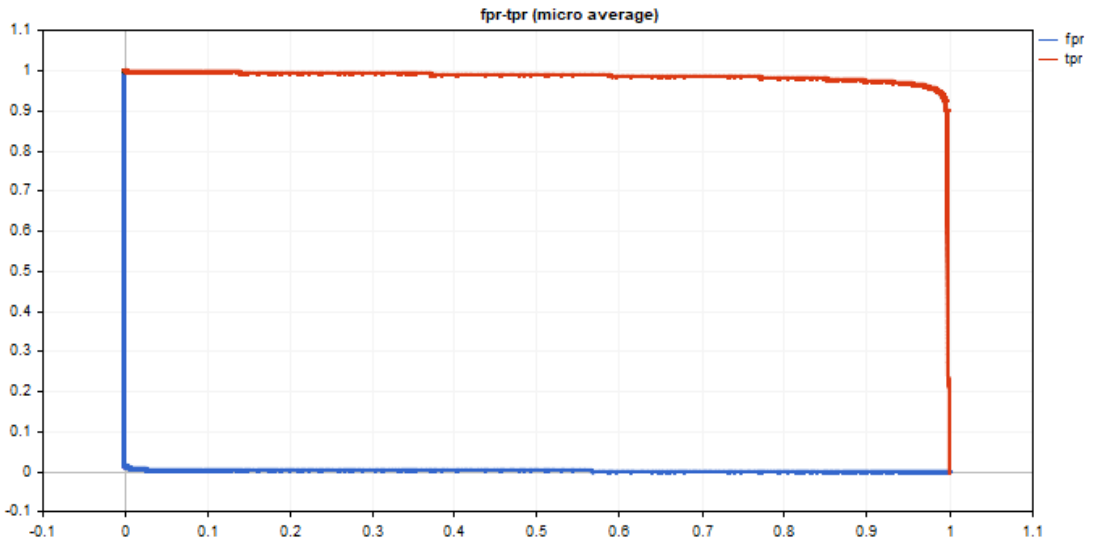
An example of plotting ROC graphs, where tpr values are plotted on the y-axis and fpr values are plotted on the x-axis. Also fpr and tpr graphs are plotted separately, with threshold values plotted on the x-axis

```
   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]
```

Dönüşüm Fonksiyonları

Bu fonksiyon grubu, verilerin bir biçimden diğerine dönüştürülmesini sağlar.

NormalizeDouble() fonksiyonunun, fiyat ifadesi için gereken kesinlik değerlerini sağladığı, özellikle not edilmelidir. Alım-satım işlemlerinde, normalize edilmemiş fiyatın kesinliği alım-satım sunucusu tarafından istenen kesinliği bir basamakla bile aşıyorsa kullanılmamalıdır.

Fonksiyon	Eylem
CharToString	Bir sembol kodunu, tek karakterlik bir dizgiye dönüştürür
DoubleToString	Sayısal bir değeri, belirtilen çözünürlükle bir metne dönüştürür
EnumToString	Herhangi tipteki bir sayım değerini dizgiye dönüştürür
NormalizeDouble	Bir kayan noktalı sayıyı, belirlenen çözünürlük ile yuvarlar
StringToDouble	Bir sayının sembol temsilini içeren bir dizgiyi, double tipli bir sayıya dönüştürür
StringToInteger	Bir sayının sembol temsilini içeren bir dizgiyi, long tipli bir sayıya dönüştürür
StringToTime	Bir zamanı veya tarihi "yyyy.mm.dd [hh:mi]" biçiminde içeren bir dizgiyi, datetime tipine dönüştürür
TimeToString	01.01.1970 tarihinden bu yana geçen saniyeleri içeren bir değeri, "yyyy.mm.dd hh:mi" biçiminde bir dizgiye dönüştürür
IntegerToString	int tipi bir değeri önceden ayarlanmış uzunlukta bir dizgiye dönüştürür
ShortToString	Sembol kodunu (unicode), tek sembollük bir dizgiye dönüştürür
ShortArrayToString	Dizinin bir bölümünü, dizgi içine kopyalar
StringToShortArray	Bir dizgiyi, ushort tipi bir dizinin seçilen kısmına sembollerle kopyalar
CharArrayToString	Bir sembol kodunu (ansi) tek sembollü bir diziye kopyalar
StringToCharArray	Unicode'dan ANSI'ye dönüştürülmüş bir dizgiyi, uchar tipi bir dizinin seçilen kısmına sembollerle kopyalar
CharArrayToStruct	uchar dizisini POD yapısına kopyalayın
StructToCharArray	POD yapısını uchar dizisine kopyala
ColorToARGB	Bir rengin ARGB temsilinin alınması için, color tipini uint tipine dönüştürür.
ColorToString	Renk değerini, "R,G,B" şeklinde bir dizgiye dönüştürür
StringToColor	Bir "R,G,B" dizgisini veya renk ismini içeren bir dizgiyi, color tipli bir değere dönüştürür
StringFormat	Önceden ayarlanmış biçime göre, sayıyı dizgiye dönüştürür

Ayrıca Bakınız

Kod Sayfasının Kullanımı

CharToString

Bir sembol kodunu, tek karakterlik bir dizgiye dönüştürür.

```
string  CharToString(
   uchar  char_code      // sembol kodunun sayısal değeri
   );
```

Parametreler

char_code

[in] ANSI sembolünün kodu.

Dönüş değeri

Bir ANSI sembollü dizgi.

Ayrıca Bakınız

StringToCharArray, ShortToString, StringGetCharacter

CharArrayToString

Uchar tipli bir dizinin bir kısmını, bir dizgiye kopyalar.

```
string  CharArrayToString(
   uchar  array[],               // dizi
   int    start=0,               // dizideki başlangıç konumu
   int    count=-1               // sembollerin sayısı
   uint    codepage=CP_ACP       // kod sayfası
   );
```

Parametreler

array[]

[in] uchar tipi dizi.

start=0

[in] Kopyalamanın başlayacağı pozisyon. ön tanımlı olarak 0 kullanılır.

count=-1

[in] Kopyalanacak dizi elemanlarının sayısı. Sonuç dizgisinin uzunluğunu tanımlar. Varsayılan değer, -1'dir, dizinin sonuna kadar veya terminal 0'a kadar anlamına gelir.

codepage=CP_ACP

[in] Kod sayfasının değeri. Uygun sabitleri sağlayan, en çok kullanılan kod sayfaları için.

Dönüş değeri

Dizgi.

Ayrıca Bakınız

StringToCharArray, ShortArrayToString, Kod Sayfasının Kullanımı

CharArrayToStruct

uchar dizisini POD yapısına kopyalayın.

```
bool  CharArrayToStruct(
   void&          struct_object,     // yapı
   const uchar&   char_array[],      // dizi
   uint           start_pos=0        // dizideki başlangıç pozisyonu
   );
```

Parametreler

struct_object

[in] Herhangi bir POD yapısına referans (sadece basit veri tiplerini içerir).

char_array[]

[in] uchar dizisi.

start_pos=0

[in] Dizideki veri kopyalamanın başladığı pozisyon.

Geri dönüş değeri

Başarılı olursa doğru, aksi takdirde yanlış olarak geri döner.

Ayrıca bakınız

StringToCharArray, ShortArrayToString, StructToCharArray, Bir Codepage'in kullanımı, FileReadStruct, Birleşimler (union), MathSwap

StructToCharArray

POD yapısını uchar dizisine kopyala.

```
bool  StructToCharArray(
   const void&  struct_object,      // yapı
   uchar&       char_array[],       // dizi
   uint         start_pos=0         // dizideki başlangıç pozisyonu
   );
```

Parametreler

struct_object

[in] Herhangi bir POD yapısına referans (sadece basit veri tiplerini içerir).

char_array[]

[in] uchar dizisi.

start_pos=0

[in] Kopyalanan verilerin ekleneceği dizideki pozisyon.

Geri dönüş değeri

Başarılı olursa doğru, aksi takdirde yanlış olarak geri döner.

Not

Kopyalama yapılırken, dinamik dizi eğer yeterli boşluk yoksa otomatik olarak (ArrayResize)'ı genişletir. Eğer dizi gerekli değere kadar genişletilemezse, fonksiyon bir hata geri döndürür.

Ayrıca bakınız

StringToCharArray, ShortArrayToString,CharArrayToStruct, Bir Codepage'in kullanımı, FileWriteStruct, Birleşimler (union), MathSwap

ColorToARGB

Fonksiyon, rengin ARGB temsilini elde etmek için, color tipini uint tipine dönüştürür. ARGB renk biçimi, bir grafiksel kaynak, metin çıktısı ve CCanvas standart kütüphane sınıfını oluşturmak için kullanılır.

```
uint  ColorToARGB(
   color  clr,          // color tipi rengi dönüştürmek için
   uchar  alpha=255     // renk saydamlığını ayarlayan alfa kanalı
   );
```

Parametreler

clr

[in] color tipi değişkendeki renk değeri.

alpha

[in] Rengi ARGB biçiminde almak için gerekli olan alfa kanalı. Bu değer, 0 ile (ön plandaki bir pikselin rengi, arka plandakini değiştirmez) 255 (arka plandaki bir pikselin rengi, tamamen üsttekinin rengini alır) arasında ayarlanabilir. Yüzdelik terimlerle renk saydamlığı şu şekilde hesaplanır (1-alpha/255)*100%. Yani alfa kanalının değeri ne kadar düşükse, rengin saydamlığı da o kadar yüksek olur.

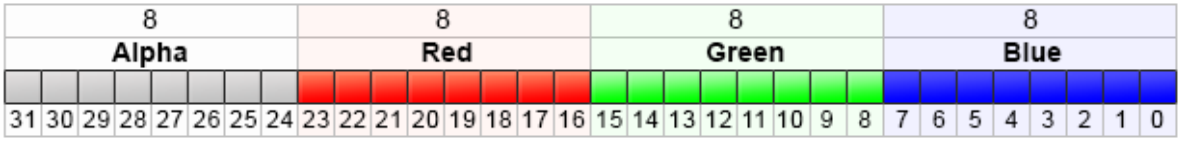
Dönüş değeri

Rengi ARGB biçiminde sunar, burada Alfa, Red, Green, Blue (alfa kanalı, kırmızı, yeşil, mavi) değerleri, dört adet uint tipi bir-baytlık seride ayarlanır.

Not

RGB, bilgisayar grafiklerinde piksel rengini tanımlamak için yaygın olarak kullanılan bir biçimdir. Temel renklerin isimleri; kırmızı (red), yeşil (green) ve mavi (blue) renk bileşenlerini ayarlamak için kullanılır. Her bir bileşen, rengin canlılığını 0 ile 255 arası (onaltılık biçimde 0x00 ile 0XFF arası) değerlerle gösteren bir bayt ile tanımlanır. Beyaz renk her rengi içerdiğinden, 0xFFFFFF şeklinde tanımlanır, yani üç bileşenin her biri 0xFF maksimum değeriyle sunulur.

Ama bazı görevler, resmin görünümünü tanımlayan renk saydamlığının belirtilmesini gerektirir, bu durumda resim, belli oranda saydamlığa sahip renklerle kaplanır. Alfa kanalı konsepti bu gibi durumlarda ortaya çıkar. RGB biçiminin bir ek bileşeni şeklinde uygulanır. ARGB biçim yapısı aşağıda gösterilmektedir.



ARGB değerleri tipik olarak, her bir hane çiftinde sırasıyla Alpha, Red, Green ve Blue kanal değerlerinin temsil edildiği onaltılık biçim ile ifade edilirler. Örneğin, 80FFFF00 rengi 50.2% mat sarıyı temsil eder. Başlangıçta 0x80, 50.2%'lik alfa değerini ayarlar, çünkü bu 0xFF değerinin 50.2%'sidir. Ardından, ilk FF çifti, kırmızının en yüksek değerini tanımlar; sonraki FF çifti, yeşil bileşenin en yüksek değerini tanımlar; son 00 çifti ise mavi bileşenin alabileceği en düşük değeri tanımlar (mavi yok). Yeşil ve kırmızı renklerin kombinasyonu sarıyı verir. Alfa kanalı kullanılmadığı

takdirde girdi, RRGGBB şeklinde altı haneli biçime dönüşecektir; bunun nedeni, alfa kanalı değerlerinin en üstte yer alan uint tipli bitlerde tutulmasıdır.

Duruma bağlı olarak, onaltılık haneler '0x' veya '#' ön eki ile yazılabilirler; örneğin, 80FFFF00, 0x80FFFF00 veya #80FFFF00.

Örnek:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- saydamlığı ayarla
   uchar alpha=0x55;  // 0x55 yani 55/255=21.6 % saydamlık
   //--- clrBlue rengini ARGB biçimine dönüştür
   PrintFormat("0x%.8X - clrBlue",clrBlue);
   PrintFormat("0x%.8X - clrBlue ARGB, alpha=0x55 (saydamlık 21.6%%)",ColorToARGB(clrB
   //--- clrGreen rengini ARGB biçimine dönüştür
   PrintFormat("0x%.8X - clrGreen",clrGreen);
   PrintFormat("0x%.8X - clrGreen ARGB, alpha=0x55 (saydamlık 21.6%%)",ColorToARGB(cl
   //--- clrRed rengini ARGB biçimine dönüştür
   PrintFormat("0x%.8X - clrRed",clrRed);
   PrintFormat("0x%.8X - clrRed ARGB, alpha=0x55 (saydamlık 21.6%%)",ColorToARGB(clrRe
  }
```

Ayrıca Bakınız

Kaynaklar, ResourceCreate(), TextOut(), color tipi, char, short, int ve long tipleri

ColorToString

Renk değerini, "R,G,B" biçimli bir dizgiye dönüştürür.

```
string  ColorToString(
   color   color_value,      // renk değeri
   bool    color_name        // renk ismini göster veya gösterme
   );
```

Parametreler

color_value

[in] color tipi değişkendeki renk değeri.

color_name

[in] Renk ismine dönüş yapılması gerektiğinin işareti - rengin ismi daha önce tanımlanmış olan renk sabitlerinden biriyle aynıysa.

Dönüş değeri

Rengin "R,G,B" biçiminde dizgi olarak temsili; burada R, G ve B değerleri, dizgiye dönüştürülen, 0-255 arası ondalık sabitlerdir. parametre color_name=true şeklinde ayarlanmışsa, renk değerini renk ismine dönüştürmeyi deneyecektir.

Örnek:

```
   string clr=ColorToString(C'0,255,0'); // yeşil renk
   Print(clr);

   clr=ColorToString(C'0,255,0',true);   // renk sabitini al
   Print(clr);
```

Ayrıca Bakınız

StringToColor, ColorToARGB

DoubleToString

Bir sayısal değeri, metin dizgisine dönüştürür.

```
string  DoubleToString(
   double  value,      // sayı
   int     digits=8    // noktadan sonraki ondalık hane sayısı
   );
```

Parametreler

value

[in] Kayan noktalı (ondalık) değer.

digits

[in] Kesinlik biçimi. Eğer *digits* değeri, 0 ve 16 arasında ise, belirtilen ondalık hane sayısıyla, bir sayının metin ifadesi elde edilir. Eğer *digits* değeri, -1 ve -16 arasında ise, belirtilen ondalık hane sayısıyla, bir sayının bilimsel biçimdeki dizgi ifadesi elde edilir. Tüm diğer durumlarda dizgi değeri, noktadan sonra 8 ondalık hane içerecektir.

Dönüş değeri

Bir sayının, belirtilen çözünürlükte sembollü ifadesini içeren dizgi.

Örnek:

```
   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));
```

Ayrıca Bakınız

NormalizeDouble, StringToDouble

EnumToString

Herhangi bir tipteki bir sayım değerini, metin biçimine dönüştürür.

```
string  EnumToString(
   any_enum  value      // herhangi bir tipte sayım değeri
   );
```

Parametreler

value

[in] Herhangi bir tipteki sayım değeri.

Dönüş değeri

Sayımın metin ifadesini içeren bir dizgi. Hata mesajını almak için GetLastError() fonksiyonunu çağırın.

Not

Bu fonksiyon, _LastError değişkeni içinde şu hata değerlerini ayarlayabilir:

- ERR_INTERNAL_ERROR - çalışma ortamı hatası
- ERR_NOT_ENOUGH_MEMORY - işlemi tamamlamak için yeterli bellek yok
- ERR_INVALID_PARAMETER - bu sayım değeri ismine izin verilmiyor

Örnek:

```
enum interval  // isimli sabitlerin sayımı
  {
   month=1,     // bir aylık zaman aralığı
   two_months,  // iki ay
   quarter,     // üç ay - bir çeyrek
   halfyear=6,  // yarım yıl
   year=12,     // bir yıl - 12 ay
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- zaman aralığını bir ay olarak ayarla
   interval period=month;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- zaman aralığını bir çeyrek (üç ay) olarak ayarla
   period=quarter;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- zaman aralığını bir yıl (12 ay) olarak ayarla
   period=year;
   Print(EnumToString(period)+"="+IntegerToString(period));

//--- emir tipinin nasıl göründüğünü kontrol et
```

```
   ENUM_ORDER_TYPE type=ORDER_TYPE_BUY;
   Print(EnumToString(type)+"="+IntegerToString(type));

//--- hatalı değerlerin nasıl göründüğünü kontrol et
   type=WRONG_VALUE;
   Print(EnumToString(type)+"="+IntegerToString(type));

// Sonuç:
// month=1
// quarter=3
// year=12
// ORDER_TYPE_BUY=0
// ENUM_ORDER_TYPE::-1=-1
  }
```

Ayrıca Bakınız

Sayımlar, Girdi değişkenleri

IntegerToString

Bu fonksiyon, tam-sayı tipli bir değeri, belirtilen uzunluktaki bir dizgiye kopyalar ve elde edilen dizgiye dönüş yapar.

```
string  IntegerToString(
   long    number,              // sayı
   int     str_len=0,           // sonuç dizgisinin uzunluğu
   ushort  fill_symbol=' '      // dolgu sembolü
   );
```

Parametreler

number

[in] Dönüştürülecek sayı.

str_len=0

[in] Dizgi uzunluğu. Sonuç dizgisinin uzunluğu belirtilenden daha büyükse, dizgi budanır. Eğer daha küçükse, dizginin soluna dolgu sembolleri eklenir.

fill_symbol=' '

[in] Dolgu sembolü. Varsayılan olarak bir boşluktur.

Dönüş değeri

Dizgi.

Ayrıca Bakınız

StringToInteger

ShortToString

Sembol kodunu (unicode) tek-sembollü dizgiye dönüştürür ve sonuç dizgisine dönüş yapar.

```
string  ShortToString(
   ushort  symbol_code       // sembol
   );
```

Parametreler

symbol_code

[in] Sembol kodu. Sembol kodunun yerine bir sembolü veya Unicode tablosundaki sembole karşılık gelen 2-baytlık onaltılık kodu içeren bir sözcük dizgisi kullanabilirsiniz.

Dönüş değeri

Dizgi.

Ayrıca Bakınız

StringToInteger

ShortArrayToString

bu fonksiyon, dizinin belli bir kısmını dizgiye kopyalar.

```
string  ShortArrayToString(
   ushort  array[],      // dizi
   int     start=0,      // dizideki başlangıç konumu
   int     count=-1      // sembol sayısı
   );
```

Parametreler

array[]

[in] ushort tipli dizi (analog wchar_t tipi).

start=0

[in] Kopyalamanın başlayacağı konum, ön tanımlı olarak - 0.

count=-1

[in] Kopyalanacak dizi elemanlarının sayısı. Sonuç dizgisinin uzunluğunu tanımlar. Varsayılan değer, -1'dir, dizinin sonuna kadar veya terminal 0'a kadar anlamına gelir.

Dönüş değeri

Dizgi.

Ayrıca Bakınız

StringToShortArray, CharArrayToString, Kod Sayfasının Kullanımı

TimeToString

01.01.1970 tarihinden bu yana geçen saniyeleri içeren bir zaman değerini, "yyyy.mm.dd hh:mi" biçimli bir dizgiye dönüştürür.

```
string  TimeToString(
   datetime  value,                          // sayı
   int       mode=TIME_DATE|TIME_MINUTES     // çıktı biçimi
   );
```

Parametreler

value

[in] 00:00 1970/01/01'den buyana geçen saniyeler cinsinden zaman.

mode=TIME_DATE|TIME_MINUTES

[in] Ek veri giriş modu. Tek veya birleştirilmiş bayraklar şeklinde olabilir:
TIME_DATE, sonucu "yyyy.mm.dd" şeklinde alır,
TIME_MINUTES, sonucu "hh:mi" şeklinde alır,
TIME_SECONDS, sonucu "hh:mi:ss" şeklinde alır.

Dönüş değeri

Dizgi.

Ayrıca Bakınız

StringToTime, TimeToStruct

NormalizeDouble

Bir kayan noktalı (ondalık) sayıyı, belirtilen kesinliğe yuvarlar.

```
double  NormalizeDouble(
   double   value,      // normalleştirilmiş sayı
   int      digits      // noktadan sonraki ondalık basamak sayısı
   );
```

Parametreler

value

[in] Kayan noktalı değer.

digits

[in] Kesinlik değeri - noktadan sonraki ondalık hane sayısı (0-8).

Dönüş değeri

Önceden ayarlanmış kesinliğe sahip double tipli değer.

Not

Zarar Durdur (StopLoss), Kar Al (TakeProfit) değerleri ve bekleyen emirlerin açılış fiyatı değerleri, Digits() fonksiyonu kullanılarak elde edilebilecek bir kesinlik değeri ile normalleştirilmelidir.

Print() fonksiyonu ile Günlüğe yapılan çıktılarda, normalleştirilmiş bir sayı için, ondalık hanelerin beklenenden çok olabileceğini not ediniz. Örneğin:

```
   double a=76.671;              // Üç ondalık haneye sahip normalleştirilmiş sayı
   Print("Print(76.671)=",a);    // Olduğu gibi çıktıla
   Print("DoubleToString(a,8)=",DoubleToString(a,8)); // Önceden ayarlanmış bir kesinl
```

komutları için, terminalde şunları elde edersiniz:

```
   DoubleToString(a,8)=76.67100000

   Print(76.671)=76.67100000000001
```

Örnek:

```
   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));
/*
  Sonuç:
```

```
   pi= 3.1415926535897931
   NormalizeDouble(pi,3)= 3.1419999999999999
   NormalizeDouble(pi,8)= 3.1415926499999998
   NormalizeDouble(pi,0)= 3.0000000000000000
*/
```

Ayrıca Bakınız

DoubleToString, Reel tipler (double, float), Tiplerin dönüşümü

StringToCharArray

Bir dizgiyi, uchar tipli bir dizinin seçilen bir kısmına yerleştirilecek şekilde, sembol bazında Unicode'dan ANSI'ye kopyalar. Kopyalanan eleman sayısına dönüş yapar.

```
int  StringToCharArray(
   string  text_string,          // kaynak dizgisi
   uchar&  array[],              // dizi
   int     start=0,              // dizideki başlangıç konumu
   int     count=-1              // sembollerin sayısı
   uint    codepage=CP_ACP       // kod sayfası
   );
```

Parametreler

text_string

[in] Kopyalanacak dizgi.

array[]

[out] uchar tipli dizgi.

start=0

[in] Kopyalamanın başlayacağı pozisyon. Varsayılan olarak - 0.

count=-1

[in] Kopyalanacak dizi elemanlarının sayısı. Sonuç dizgisinin uzunluğunu tanımlar. Varsayılan değer, -1'dir, dizinin sonuna kadar veya terminal 0'a kadar anlamına gelir. Ayrıca 'terminal 0' da diziye kopyalanacaktır, bu durumda dinamik dizinin büyüklüğü, dizgi büyüklüğü için gerekli olan alan için artırılabilir. Eğer dinamik dizinin büyüklüğü dizgi büyüklüğünü aşarsa, dizi büyüklüğü azaltılmayacaktır.

codepage=CP_ACP

[in] Kod sayfasının değeri. Uygun sabitleri sağlayan, en çok kullanılan kod sayfaları için.

Dönüş değeri

Kopyalanan eleman sayısı.

Ayrıca Bakınız

CharArrayToString, StringToShortArray, Kod Sayfasının Kullanımı

StringToColor

"R,G,B" dizgisini veya bir renk ismini içeren dizgiyi color tipi değere dönüştürür.

```
color  StringToColor(
   string  color_string      // rengin dizgi şeklinde ifadesi
   );
```

Parametreler

color_string

[in] "R,G,B" tipli bir rengin veya ön tanımlı Web-renklerinden birinin isminin dizgi şeklindeki temsili.

Dönüş değeri

Renk değeri.

Örnek:

```
   color str_color=StringToColor("0,127,0");
   Print(str_color);
   Print((string)str_color);
//--- rengi biraz değiştir
   str_color=StringToColor("0,128,0");
   Print(str_color);
   Print((string)str_color);
```

Ayrıca Bakınız

ColorToString, ColorToARGB

StringToDouble

Bir sayının sembol temsilini içeren bir dizgiyi, double tipi bir sayıya dönüştürür.

```
double  StringToDouble(
   string  value      // dizgi
   );
```

Parametreler

value

[in] Bir sayının sembol temsilini içeren dizgi.

Dönüş değeri

double tipli değer.

Ayrıca Bakınız

NormalizeDouble, Reel tipler (double, float), Tiplerin dönüşümü

StringToInteger

Bir sayının sembol temsilini içeren bir dizgiyi, long (tamsayı) tipi bir sayıya dönüştürür.

```
long  StringToInteger(
   string  value      // dizgi
   );
```

Parametreler

value

[in] Bir sayıyı içeren dizgi.

Dönüş değeri

long tipli değer.

Ayrıca Bakınız

IntegerToString, Reel tipler (double, float), Tiplerin dönüşümü

StringToShortArray

Bu fonksiyon, bir dizgiyi sembol bazında, ushort tipli bir dizinin belirtilen alanına kopyalar ve ardından, kopyalanan eleman sayısına dönüş yapar.

```
int  StringToShortArray(
   string  text_string,      // kaynak dizgisi
   ushort& array[],          // dizi
   int     start=0,          // dizideki başlangıç konumu
   int     count=-1          // sembollerin sayısı
   );
```

Parametreler

text_string

[in] Kopyalanacak dizgi

array[]

[out] ushort tipi dizi (analog wchar_t tipi).

start=0

[in] Kopyalamanın başlayacağı konum. Varsayılan olarak - 0.

count=-1

[in] Kopyalanacak dizi elemanlarının sayısı. Sonuç dizgisinin uzunluğunu tanımlar. Ön tanımlı değer -1'dir, yani dizinin sonuna kadar veya terminal 0'a kadar kopyalama yapılacağı anlamına gelir. "Terminal 0" da ayrıca diziye kopyalanacaktır, bu durumda dinamik dizinin büyüklüğü, dizgi büyüklüğü için gerekli olan alan için artırılabilir. Eğer dinamik dizinin büyüklüğü dizgi büyüklüğünü aşarsa, dizi büyüklüğü azaltılmayacaktır.

Dönüş değeri

Kopyalanan eleman sayısı.

Ayrıca Bakınız

ShortArrayToString, StringToCharArray, Kod Sayfasının Kullanımı

StringToTime

"yyyy.mm.dd [hh: mi]" biçimindeki saat ve/veya tarihi içeren dizgeyi datetime tip numarasına dönüştürür.

```
datetime  StringToTime(
   const string  time_string      // tarih dizgesi
   );
```

Parametreler

time_string

[in] Belirtilen biçimlerden birindeki dizge:

- "yyyy.mm.dd [hh:mi]"
- "yyyy.mm.dd [hh:mi:ss]"
- "yyyymmdd [hh:mi:ss]"
- "yyyymmdd [hhmiss]"
- "yyyy/mm/dd [hh:mi:ss]"
- "yyyy-mm-dd [hh:mi:ss]"

Geri dönüş değeri

01.01.1970 tarihinden beri geçen saniye sayısını içeren datetime tip değeri.

Not

Tarih ve saat arasındaki tüm boşluk ve çizelgeleme karakterlerinin bölümü, StringToTime() fonksiyonunu çağırmadan önce ilave *time_string* işlemini önlemek için tek bir boşluk olarak kabul edilir.

Ayrıca bakınız

TimeToString, TimeToStruct

StringFormat

Bu fonksiyon alınan parametreleri biçimlendirir ve bir dizgiye dönüştürür.

```
string  StringFormat(
   string  format,      // Biçimlendirme açıklamasını içeren dizgi
   ...     ...          // parametreler
   );
```

Parametreler

formatında olmalıdır.

[in] Biçimlendirme yöntemini içeren dizgi. Biçimlendirme kuralları, PrintFormat fonksiyonundaki gibidir.

...

[in] Virgülle ayrılmış parametreler.

Dönüş değeri

Dizgi.

Örnek:

```
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)
*/
  }
```

Ayrıca Bakınız

PrintFormat, DoubleToString,ColorToString, TimeToString

Matematiksel fonksiyonlar

Matematiksel ve trigonometrik fonksiyonlar kümesi.

Matematik fonksiyonları başlangıçta skaler değerler üzerinde matematik işlemleri gerçekleştirmek için tasarlanmıştır. Artık bu fonksiyonların çoğu matrisler ve vektörlerle kullanılabilir. İlgili fonksiyonlar şunlardır: MathAbs, MathArccos, MathArcsin, MathArctan, MathCeil, MathCos, MathExp, MathFloor, MathLog, MathLog10, MathMod, MathPow, MathRound, MathSin, MathSqrt, MathTan, MathExpm1, MathLog1p, MathArccosh, MathArcsinh, MathArctanh, MathCosh, MathSinh, ve MathTanh. Burada, matrisler veya vektörler öğe bazında işlenir. Örnek:

```
//---
  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]]
  */
```

MathMod ve MathPowiçin ikinci öğe uygun boyutta bir skaler veya matris/vektör olabilir.

Fonksiyon	Eylem
MathAbs	Belirtilen sayısal değerin mutlak değerine dönüş yapar
MathArccos	Bir x değişkeninin ark kosinüs değerine, radyan cinsinden dönüş yapar
MathArcsin	Bir x değişkeninin ark sinüs değerine, radyan cinsinden dönüş yapar
MathArctan	Bir x değişkeninin ark tanjant değerine, radyan cinsinden dönüş yapar
MathClassify	Reel sayının tipini geri döndürür
MathCeil	Nümerik bir değer için en yakın büyük tamsayı değerine dönüş yapar
MathCos	Bir sayının kosinüsüne dönüş yapar
MathExp	Bir sayının eksponentine dönüş yapar
MathFloor	Nümerik bir değer için en yakın küçük tamsayı değerine dönüş yapar
MathLog	Sayının normal logaritmasına dönüş yapar
MathLog10	Bir sayının 10 tabanındaki logaritmasına dönüş yapar
MathMax	İki sayısal sayı arasındaki en büyük değere dönüş yapar

Fonksiyon	Eylem
MathMin	İki sayısal sayı arasındaki en küçük değere dönüş yapar
MathMod	İki sayının bölümünden kalan reel sayıya dönüş yapar
MathPow	Bir sayının kuvvetini alır
MathRand	0 ile 32767 arasında bir (psuedo) rassal sayıya dönüş yapar
MathRound	Değeri, en yakın tamsayıya yuvarlar
MathSin	Bir sayının sinüsüne dönüş yapar
MathSqrt	Sayının karekköküne dönüş yapar
MathSrand	Bir dizi (psuedo) rassal sayı oluşturmak için başlangıç değeri ayarlar
MathTan	Bir sayının tanjantını alır
MathIsValidNumber	Bir reel sayının doğruluğunu kontrol eder
MathExpm1	MathExp(x)-1 ifadesinin değerine dönüş yapar
MathLog1p	MathLog(1+x) ifadesinin değerine dönüş yapar
MathArccosh	Hiperbolik ark kosinüs değerine dönüş yapar
MathArcsinh	Hiperbolik ark sinüs değerine dönüş yapar
MathArctanh	Hiperbolik ark tanjant değerine dönüş yapar
MathCosh	Hiperbolik kosinüs değerine dönüş yapar
MathSinh	Hiperbolik sinüs değerine dönüş yapar
MathTanh	Hiperbolik tanjant değerine dönüş yapar
MathSwap	ushort/uint/ushort değerindeki bayt sırasını değiştirin

MathAbs

Fonksiyon, belirtilen sayısal değerin mutlak değerine dönüş yapar.

```
double  MathAbs(
   double  value      // sayısal değer
   );
```

Parametreler

value

[in] Nümerik değer.

Dönüş değeri

Sıfıra eşit veya sıfırdan büyük double tipli değer.

Not

MathAbs() fonksiyonu yerine fabs() kullanabilirsiniz.

MathArccos

Fonksiyon, bir x değişkeninin ark kosinüsüne, 0 ile π aralığında radyan cinsinden dönüş yapar.

```
double  MathArccos(
   double  val     // -1<val<1
   );
```

Parametreler

val

[in] Ark kosinüsü hesaplanacak 'val' değeri, -1 ile 1 arasındadır.

Dönüş değeri

Sayının radyan cinsinden ark kosinüs değeri. Eğer val değeri -1'den küçük veya 1'den büyükse, fonksiyon NaN (belirsiz değer) dönüşü yapar.

Not

MathArccos() fonksiyonunun yerine acos() kullanabilirsiniz.

Ayrıca Bakınız

Reel tipler (double, float)

MathArcsin

Fonksiyon, bir x değişkeninin ark kosinüsüne -π/2 ile π/2 aralığında radyan cinsinden dönüş yapar.

```
double  MathArcsin(
   double  val      // -1<value<1
   );
```

Parametreler

val

[in] Ark sinüsü hesaplanacak 'val' değeri, -1 ile 1 arasındadır.

Dönüş değeri

'val' sayısının -π/2 ile π/2 aralığında radyan cinsinden ark sinüs değeri. Eğer 'val' değeri -1'den küçük veya 1'den büyükse, fonksiyon NaN (belirsiz değer) dönüşü yapar.

Not

MathArcsin() fonksiyonunun yerine asin() kullanabilirsiniz.

Ayrıca Bakınız

Reel tipler (double, float)

MathArctan

Fonksiyon, bir x değişkeninin ark tanjant değerine dönüş yapar. Eğer x, sıfıra eşitse, fonksiyon 0 dönüşü yapar.

```
double  MathArctan(
   double  value      // tanjant
   );
```

Parametreler

value

[in] Tanjantı istenen sayı.

Dönüş değeri

MathArctan fonksiyonu -π/2 ile π/2 arasında radyan cinsinden bir değere dönüş yapar.

Not

MathArctan() fonksiyonunun yerine atan() kullanabilirsiniz.

MathArctan2

Tanjantı, belirtilen iki sayının bölümü olan açıyı (radyan cinsinden) geri döndürür.

```
double MathArctan2(
 double y     // Bir noktanın y koordinatı
 double x     // Bir noktanın x koordinatı
  );
```

Parametreler

y

[in] Y koordinatı değeri.

x

[in] X koordinatı değeri.

Geri dönüş değeri

MathArctan2, -π ile π radyanları arasında bir θ açısı geri döndürür, böylece MathTan(θ) = y/x olur.

Lütfen aşağıdaki ifadelere dikkat edin:

- 1. bölgedeki (x, y) için, 0 < θ < π/2
- 2. bölgedeki (x, y) için, π/2 < θ ≤ π
- 3. bölgedeki (x, y) için, -π < θ < -π/2
- 4. bölgedeki (x, y) için, -π/2 < θ < 0

Bölgelerin sınırları üzerindeki noktalar için, geri dönüş değeri aşağıdaki gibidir:

- Eğer y = 0 ise ve x negatif değilse, θ = 0.
- Eğer y = 0 ise ve x negatifse, θ = π.
- Eğer y pozitif ve x = 0 ise, θ = π/2.
- Eğer y negatif ve x = 0 ise, θ = -π/2.
- Eğer y ve x değerlerinin ikisi de 0 ise, θ = 0.

Not

MathArctan2() fonksiyonu yerine atan2() fonksiyonunu kullanabilirsiniz.

MathClassify

Reel sayının tipini belirler ve ENUM_FP_CLASS numaralandırmasından değer cinsinde sonuç geri döndürür.

```
ENUM_FP_CLASS  MathClassify(
   double  value      // reel sayı
   );
```

Parametreler

value

[in] Kontrol edilecek reel sayı

Geri dönüş değeri

ENUM_FP_CLASS numaralandırmasından bir değer

ENUM_FP_CLASS

Tanımlayıcı	Açıklama
FP_SUBNORMAL	Gösterilebilen en küçük normal sayı olan DBL_MIN'den (2.2250738585072014e-308) daha sıfıra yakın olan bir normal altı sayı
FP_NORMAL	2.2250738585072014e-308 ile 1.7976931348623158e+308 arasında bir normal sayı
FP_ZERO	Artı veya eksi sıfır
FP_INFINITE	Karşılık gelen tiple gösterilemeyen bir sayı - artı veya eksi sonsuz
FP_NAN	Bir sayı değil

Örnek:

```
//+------------------------------------------------------------------+
//| 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
*/
//+------------------------------------------------------------------+
```

Ayrıca bakınız

Reel tipler (double, float), MathIsValidNumber

MathCeil

Fonksiyon, sayısal bir değer için en yakın büyük tamsayı değerine dönüş yapar.

```
double  MathCeil(
   double  val      // sayı
   );
```

Parametreler

val

[in] Nümerik değer.

Dönüş değeri

'val' değerine eşit veya ondan daha büyük olan en küçük tamsayı değeri.

Not

MathCeil() fonksiyonunun yerine ceil() kullanabilirsiniz.

MathCos

Bir sayının kosinüsüne dönüş yapar.

```
double  MathCos(
   double  value      // sayı
   );
```

Parametreler

value

[in] Radyan cinsinden açı değeri.

Dönüş değeri

-1 ile 1 arasında, double tipli bir değer.

Not

MathCos() fonksiyonunun yerine cos() fonksiyonunu kullanabilirsiniz.

MathExp

Bu fonksiyon, e sayısının d üssüne dönüş yapar.

```
double  MathExp(
   double  value      // e sayısı için üs değeri
   );
```

Parametreler

value

[in] Üssü belirleyen bir sayı.

Dönüş değeri

double tipli bir sayı. Aşım durumunda fonksiyon INF (sonsuz), mertebe kaybı durumunda ise 0 dönüşü yapar.

Not

MathExp() fonksiyonunun yerine exp() kullanabilirsiniz.

Ayrıca Bakınız

Reel tipler (double, float)

MathFloor

Sayısal bir değer için en yakın küçük tamsayı değerine dönüş yapar.

```
double  MathFloor(
   double  val     // sayı
   );
```

Parametreler

val

[in] Nümerik değer.

Dönüş değeri

'val' değerine eşit veya ondan daha küçük olan en büyük tamsayı değeri.

Not

MathFloor() fonksiyonunun yerine floor() kullanabilirsiniz.

MathLog

Belirtilen sayının normal logaritmasına dönüş yapar.

```
double  MathLog(
   double  val      // logaritması alınacak olan değer
   );
```

Parametreler

val

[in] Logaritması alınacak değer.

Dönüş değeri

Başarı durumunda 'val' değişkeninin doğal logaritması. 'val' negatifse, fonksiyon NaN (tanımsız değer) dönüşü yapar. 'val' değeri 0 ise, INF (sonsuz) dönüşü yapar.

Not

MathLog() yerine log() fonksiyonunu kullanabilirsiniz.

Ayrıca Bakınız

Reel tipler (double, float)

MathLog

Bir sayının 10 tabanındaki logaritmasına dönüş yapar.

```
double  MathLog10(
   double  val      // logaritması alınacak sayı
   );
```

Parametreler

val

[in] Logaritması hesaplanacak sayısal değer.

Dönüş değeri

Başarı durumunda, sayının logaritmasına dönüş yapar. 'val' negatifse, fonksiyon NaN (tanımsız değer) dönüşü yapar. 'val' değeri 0 ise, INF (sonsuz) dönüşü yapar.

Not

MathLog10() fonksiyonunun yerine log10() kullanabilirsiniz.

Ayrıca Bakınız

Reel tipler (double, float)

MathMax

Fonksiyon, iki değer arasındaki en büyük değere dönüş yapar.

```
double  MathMax(
   double  value1,      // ilk değer
   double  value2       // ikinci değer
   );
```

Parametreler

value1

[in] İlk sayısal değer.

value2

[in] İkinci sayısal değer.

Dönüş değeri

İki değer arasında en büyük olana dönüş yapar.

Not

MathMax() fonksiyonunun yerine fmax() kullanabilirsiniz. fmax(), fmin(), MathMax(), MathMin() fonksiyonları, double tip dönüşümü yapılmaksızın tamsayı tipleriyle de kullanılabilir.

Fonksiyon farklı tiplerde parametreler ile kullanıldığında, daha küçük tipli parametre, otomatik olarak büyük tipe dönüştürülür. Dönüş değerinin tipi büyük tipe karşılık gelir.

Aynı tipteki veriler için dönüşüm gerçekleştirilmez.

MathMin

Fonksiyon, iki değer arasındaki en küçük değere dönüş yapar.

```
double  MathMin(
   double  value1,      // ilk değer
   double  value2       // ikinci değer
   );
```

Parametreler

value1

[in] İlk sayısal değer.

value2

[in] İkinci sayısal değer.

Dönüş değeri

İki değer arasında en büyük olana dönüş yapar.

Not

MathMin() fonksiyonunun yerine fmin() kullanabilirsiniz. fmax(), fmin(), MathMax(), MathMin() fonksiyonları, double tip dönüşümü yapılmaksızın tamsayı tipleriyle de kullanılabilir.

Fonksiyon farklı tiplerde parametreler ile kullanıldığında, daha küçük tipli parametre, otomatik olarak büyük tipe dönüştürülür. Dönüş değerinin tipi büyük tipe karşılık gelir.

Aynı tipteki veriler için dönüşüm gerçekleştirilmez.

MathMod

İki sayının bölümünden kalan reel sayıya dönüş yapar.

```
double  MathMod(
   double  value,       // bölünen değer
   double  value2       // bölen değer
   );
```

Parametreler

value

[in] Bölünen değer.

value2

[in] Bölen değer.

Dönüş değeri

MathMod fonksiyonu "val / y" bölümünün reel değerini "val = i * y + f" şeklinde hesaplar. Burada i bir tamsayıdır, f 'val' ile aynı işarete sahiptir ve mutlak değeri y'nin mutlak değerinden küçüktür

Not

MathMod() fonksiyonunun yerine fmod() kullanabilirsiniz.

MathPow

Fonksiyon, bir sayının belirtilen kuvvetini alır.

```
double  MathPow(
   double  base,          // sayı
   double  exponent       // üs değeri
   );
```

Parametreler

base

[in] Üssü alınacak sayı.

exponent

[in] Üs değeri.

Dönüş değeri

Sayının kuvveti hesaplandıktan sonraki değeri.

Not

MathPow() fonksiyonunun yerine pow() kullanabilirsiniz.

MathRand

0 ile 32767 arasında bir (psuedo) rassal tamsayıya dönüş yapar.

```
int  MathRand();
```

Dönüş değeri

0 ile 32767 arasında bir tamsayı değeri.

Not

Fonksiyonun ilk kullanımından önce, (psuedo) rassal sayı üretecinin başlangıç pozisyonunu ayarlamak amacıyla MathSrand fonksiyonunun çağrılması gerekir.

Not

MathRand() yerine rand() fonksiyonunu kullanabilirsiniz.

MathRound

Belirtilen değere en yakın olan tamsayı değerine dönüş yapar.

```
double  MathRound(
   double  value      // yuvarlanması istenen değer
   );
```

Parametreler

value

[in] Yuvarlamadan önceki değer.

Dönüş değeri

Belirtilen değere en yakın tamsayı.

Not

MathRound() fonksiyonunun yerine round() fonksiyonunu kullanabilirsiniz.

MathSin

Belirtilen sayının sinüsüne dönüş yapar.

```
double  MathSin(
   double  value      // radyan cinsinden argüman
   );
```

Parametreler

value

[in] Radyan cinsinden açı değeri.

Dönüş değeri

Sayının (açının) radyan cinsinden sinüsü. -1 ile 1 arasında bir değere dönüş yapar.

Not

MathSin() fonksiyonunun yerine sin() kullanabilirsiniz.

MathSqrt

Belirtilen sayının kareköküne dönüş yapar.

```
double  MathSqrt(
   double  value      // pozitif sayı
   );
```

Parametreler

value

[in] Pozitif sayısal değer.

Dönüş değeri

Değerin karekökü. Eğer değer negatifse, MathSqrt fonksiyonu NaN (tanımsız değer) dönüşü yapar.

Not

MathSqrt() yerine sqrt() fonksiyonunu da kullanabilirsiniz.

Ayrıca Bakınız

Reel tipler (double, float)

MathSrand

Bir dizi (psuedo) rassal sayı oluşturmak için başlangıç değeri ayarlar.

```
void  MathSrand(
   int  seed      // başlangıç sayısı
   );
```

Parametreler

seed

[in] Rassal sayı dizisi için başlangıç değeri.

Dönüş değeri

Dönüş değeri yok.

Not

MathRand() fonksiyonu bir dizi (psuedo) rassal sayı oluşturmak için kullanılır. MathSrand() fonksiyonunun belli bir değerle başlatılmasının sonucunda her zaman aynı (pseudo) rassal sayı dizisi elde edilir.

Tekrarlayan dizilerden kaçınmak için MathSrand(GetTickCount()) çağrısını kullanın; GetTickCount() değeri işletim sisteminin başlatılması anından itibaren artmaya başlar ve 49 gün boyunca, sisteme gömülü milisaniye sayacının sınırları aşılana kadar tekrar etmez. MathSrand(TimeCurrent()) çağrısı ise uygun bir çözüm değildir, çünkü TimeCurrent() son tik zamanını verir - ki bu değer, bir hafta gibi daha kısa bir süre içinde tekrar edebilir.

Rassal sayı üretecinin, göstergeler ve Uzman Danışmanlar için MathSrand() fonksiyonu ile OnInit() işleyicisi içinde başlatılması önerilir, böylece OnTick() veya OnCalculate() içinde gerçekleşecek tekrar-başlatma durumu önlenebilir.

MathSrand() fonksiyonunun yerine srand() fonksiyonunu da kullanabilirsiniz.

Örnek:

```
#property description "Bu gösterge, birbirine yakın büyüklüğe sağip (toplamları sözkor
#property description "her hangi birinin toplamı çok fazla etkilemeyeceği) rassal değ
#property description "yeterince büyük bir sayıya ulaşmaları durumunda, dağılımlarının
#property description "yakınsayacağını ifade eden Merkezi Limit teoremini gösterir."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- Grafiksel çizim özellikleri
#property indicator_label1  "Label"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrRoyalBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  5
//--- Bir girdi değişkeni
input int      sample_number=10;
//--- Dağılımın çizilmesi için bir gösterge tamponu
```

```
double         LabelBuffer[];
//--- Tik sayacı
double         ticks_counter;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- Gösterge tamponunun dizi ile bağlanması
   SetIndexBuffer(0,LabelBuffer,INDICATOR_DATA);
//--- göstergenin erişim yönünü şimdiki zamandan geçmişe doğru ayarla
   ArraySetAsSeries(LabelBuffer,true);
//--- rassal sayı üretecini başlat
   MathSrand(GetTickCount());
//--- Tik sayacını başlat
   ticks_counter=0;
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- Sıfır sayacı için, gösterge tamponunu sıfırla
   if(ticks_counter==0) ArrayInitialize(LabelBuffer,0);
//--- Sayacı artır
   ticks_counter++;
//--- Periyodik olarak tik sayacını sıfırlamalıyız, bu şekilde dağılımı görebiliriz
   if(ticks_counter>100)
     {
      Print("Gösterge değerleri sıfırlandı, hücrelerin doldurulmasına yeniden başlaya
      ticks_counter=0;
     }
//--- 0 ile 7 arasındaki üç sayının toplamı şeklinde rassal değerler örneğini al
   for(int i=0;i<sample_number;i++)
     {
      //--- Rassal sayının, başka üç sayının toplamı olarak alınacağı hücrenin indisi
      int rand_index=0;
      //--- 0 ile 7 arasında üç adet rassal sayı seç
      for(int k=0;k<3;k++)
        {
```

```
         //--- 7 ile bölümden kalan; 0 ile 6 arası bir değere dönüş yapar
         rand_index+=MathRand()%7;
        }
      //--- rand_index hücre numarası değerini 1 artır
      LabelBuffer[rand_index]++;
     }
//--- OnCalculate() işleyicisinden çık
   return(rates_total);
  }
```

MathTan

Fonksiyon, belirtilen sayının tanjantına dönüş yapar

```
double  MathTan(
   double  rad      // radyan cinsi argüman
   );
```

Parametreler

rad

[in] Radyan cinsinden açı değeri.

Dönüş değeri

'rad' değerinin tanjantı. 'rad' değeri 263'e eşit veya ondan büyük veya -263'e eşit veya ondan küçükse, sonuçta bir anlamlılık kaybı gerçekleşir. Bu durumda fonksiyon, tanımsız değere dönüş yapar.

Not

MathTan() fonksiyonunun yerine tan() kullanabilirsiniz.

Ayrıca Bakınız

Reel tipler (double, float)

MathIsValidNumber

Bir reel sayının doğruluğunu kontrol eder.

```
bool  MathIsValidNumber(
   double  number      // kontrol edilecek sayı
   );
```

Parametreler

number

[in] Kontrol edilen sayısal değer.

Dönüş değeri

Kontrol edilen değer kabul edilebilir bir sayı ise 'true' dönüşü yapar. Kontrol edilen sayı artı veya eksi sonsuz ise veya belirsiz değer (NaN) ise, 'false' dönüşü yapar.

Örnek:

```
   double abnormal=MathArcsin(2.0);
   if(!MathIsValidNumber(abnormal)) Print("Dikkat! MathArcsin(2.0) = ",abnormal);
```

Ayrıca Bakınız

Reel tipler (double, float)

MathExpm1

MathExp(x)-1 ifadesinin değerine dönüş yapar.

```
double  MathExpm1(
   double  value      // e sayısı için üs değeri
   );
```

Parametreler

value

[in] Üs değeri.

Dönüş Değeri

double tipli değer. Değerin fazla yüksek olması durumunda fonksiyon INF (sonsuz) dönüşü yapar, değer çok düşükse MathExpm1 '0' dönüşü yapar.

Not

Sıfıra yakın *x* değerleri için, MathExpm1(x) fonksiyonu MathExp(x)-1 fonksiyonuna göre çok daha doğru sonuç verir.

MathExpm1() fonksiyonunun yerine expm1() fonksiyonunu da kullanabilirsiniz.

Ayrıca bakınız

Reel tipler (double, float)

MathLog1p

MathLog(1+x) ifadesinin değerine dönüş yapar.

```
double  MathLog1p(
   double  value      // logaritması alınacak değer
   );
```

Parametreler

value

[in] Logaritması alınacak değer.

Dönüş Değeri

Başarı durumunda (value + 1) değerinin doğal logaritması. value < -1 ise, fonksiyon NaN (tanımsız) dönüşü yapar. value, -1 ise fonksiyon INF (sonsuz) dönüşü yapar.

Not

x değeri sıfıra yakınsa, MathLog1p(x) fonksiyonu MathLog(1+x) fonksiyonuna göre daha doğru sonuç verir.

MathLog1p() fonksiyonunun yerine log1p() fonksiyonunu da kullanabilirsiniz.

Ayrıca bakınız

Reel tipler (double, float)

MathArccosh

Hiperbolik ark kosinüs değerine dönüş yapar.

```
double  MathArccosh(
   double  value     // 1 <= değer < ∞
   );
```

Parametreler

value

[in] Hiperbolik ark kosinüs hesaplaması için kullanılacak sayı.

Dönüş Değeri

Sayının hiperbolik ark kosinüs değeri. Değer +1'den küçükse, NaN (tanımsız değer) dönüşü yapar.

Not

MathArccosh() fonksiyonunun yerine acosh() fonksiyonunu kullanabilirsiniz.

Ayrıca bakınız

Reel tipler (double, float)

MathArcsinh

Hiperbolik ark sinüs değerine dönüş yapar.

```
double  MathArcsinh(
   double  value      // -∞ < değer < +∞
   );
```

Parametreler

val

[in] Hiperbolik ark sinüs hesaplaması için kullanılacak sayı.

Dönüş Değeri

Sayının hiperbolik ark sinüs değeri.

Not

MathArcsinh() fonksiyonunun yerine asinh() fonksiyonunu da kullanabilirsiniz.

Ayrıca bakınız

Reel tipler (double, float)

MathArctanh

Hiperbolik ark tanjant değerine dönüş yapar.

```
double  MathArctanh(
   double  value      // -1 < değer < 1
   );
```

Parametreler

value

[in] (-1, 1) aralığında tanjant değerini temsil eden bir sayı.

Dönüş Değeri

Sayının hiperbolik ark tanjant değeri.

Not

MathArctanh() fonksiyonunun yerine atanh() fonksiyonunu da kullanabilirsiniz.

MathCosh

Hiperbolik kosinüs değerine dönüş yapar.

```
double  MathCosh(
   double  value      // sayı
   );
```

Parametreler

value

[in] Değer.

Dönüş Değeri

Sayının hiperbolik kosinüs değeri, (1,∞) aralığında bir değer.

Not

MathArccosh() fonksiyonunun yerine acosh() fonksiyonunu da kullanabilirsiniz.

MathSinh

Hiperbolik sinüs değerine dönüş yapar.

```
double  MathSinh(
   double  value      // sayı
   );
```

Parametreler

value

[in] Değer.

Dönüş Değeri

Sayının hiperbolik sinüs değeri.

Not

MathSinh() fonksiyonunun yerine sinh() fonksiyonunu da kullanabilirsiniz.

MathTanh

Hiperbolik tanjant değerine dönüş yapar.

```
double  MathTanh(
   double  value      // sayı
   );
```

Parametreler

value

[in] Değer.

Dönüş Değeri

Sayının hiperbolik tanjant değeri, (-1,1) aralığında bir değer.

Not

MathTanh() fonksiyonunun yerine atanh() fonksiyonunu da kullanabilirsiniz.

Ayrıca bakınız

Reel tipler (double, float)

MathSwap

ushort değerindeki bayt sırasını değiştirin.

```
ushort  MathSwap(
   ushort  value      // değer
   );
```

Parametreler

value

[in] Bayt sırasını değiştirmek için değer.

Geri dönüş değeri

Ters bayt sırası ile ushort değeri.

MathSwap

uint değerindeki bayt sırasını değiştirin.

```
uint  MathSwap(
   uint   value      // değer
   );
```

Parametreler

value

[in] Bayt sırasını değiştirmek için değer.

Geri dönüş değeri

Ters bayt sırası ile uint değeri.

MathSwap

ulong değerindeki bayt sırasını değiştirin.

```
ulong  MathSwap(
   ulong  value      // değer
   );
```

Parametreler

value

[in] Bayt sırasını değiştirmek için değer.

Geri dönüş değeri

Ters bayt sırası ile ulong değeri.

Dizgi Fonksiyonları

Bu fonksiyon grubu string tipli verilerle çalışmak için tasarlanmıştır.

Fonksiyon	Eylem
StringAdd	Bir dizgiyi başka bir dizginin sonuna ekler
StringBufferLen	Dizgi için tahsis edilmiş olan tamponun (ara-bellek) boyutuna dönüş yapar
StringCompare	İki dizgiyi karşılaştırır. Birinci dizgi ikincisinden büyük ise 1; dizgiler eşitse 0; ilk dizgi ikincisinden küçükse -1 dönüşü yapar.
StringConcatenate	Geçirilen parametrelerden oluşan bir dizgiyi şekillendirir
StringFill	Belirtilen bir dizgiyi, seçilen sembollerle doldurur
StringFind	Dizgi içinde, bir alt-dizgiyi arar
StringGetCharacter	Dizgi içerisinde, belirtilen konumdaki bir sayının değerine dönüş yapar
StringInit	Belirtilen sembollerle bir dizgiyi başlatır ve belirtilen dizgi uzunluğunu verir
StringLen	Bir dizgi içindeki sembollerin sayısını verir
StringSetLength	Bir dizge için belirtilen bir uzunluk (karakter olarak) ayarlar
StringReplace	Dizgi içinde bulunan belirli alt-dizgileri, belirtilen sembol dizileriyle değiştirir
StringReserve	Bellekteki bir dizge için belirtilen boyutta bir tampon ayırır
StringSetCharacter	Belirtilen konumundaki sembol değeri değiştirilmiş bir dizginin kopyasına dönüş yapar
StringSplit	Belirtilen ayracı kullanarak, belirtilen bir dizgiden alt-dizgiler elde eder ve bu alt-dizgilerin sayısına dönüş yapar
StringSubstr	Bir metnin belirtilen konumundan bir alt-dizgiyi ayıklar
StringToLower	Bir dizginin tüm sembollerini, konuma göre küçük karaktere çevirir
StringToUpper	Bir dizginin tüm sembollerini, konuma göre büyük karakterlere çevirir
StringTrimLeft	Dizginin sol tarafındaki satır atlama karakterlerini, boşlukları ve sekmeleri kırpar
StringTrimRight	Dizginin sağ tarafındaki satır atlama karakterlerini, boşlukları ve sekmeleri keser

StringAdd

Fonksiyon, bir dizginin sonuna bir alt-dizgi ekler.

```
bool  StringAdd(
   string&  string_var,        // ekleme yapmak istediğimiz dizgi
   string   add_substring      // eklenecek dizgi
   );
```

Parametreler

string_var

[in][out] Ekleme yapmak istediğimiz dizgi.

add_substring

[in] Kaynak dizginin sonuna eklenecek olan dizgi.

Dönüş değeri

Başarılı sonuç durumunda 'true', aksi durumda ise 'false' dönüşü yapar. Bir hata kodu almak için, GetLastError() fonksiyonunu çağırın.

Örnek:

```
void OnStart()
  {
   long length=1000000;
   string a="a",b="b",c;
//--- birinci yöntem
   uint start=GetTickCount(),stop;
   long i;
   for(i=0;i<length;i++)
     {
      c=a+b;
     }
   stop=GetTickCount();
   Print("'c = a + b' için kullanılan zaman = ",(stop-start)," milisaniye, i = ",i);

//--- ikinci yöntem
   start=GetTickCount();
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   stop=GetTickCount();
   Print("'StringAdd(a,b)' için kullanılan zaman = ",(stop-start)," milisaniye, i = ",

//--- üçüncü yöntem
   start=GetTickCount();
   a="a"; // a değişkenini yeniden başlat
   for(i=0;i<length;i++)
```

```
      {
       StringConcatenate(c,a,b);
      }
   stop=GetTickCount();
   Print("'StringConcatenate(c,a,b)' için harcanan zaman = ",(stop-start)," milisaniye
  }
```

Ayrıca Bakınız

StringConcatenate, StringSplit, StringSubstr

StringBufferLen

Dizgi için tahsis edilmiş tamponunun (ara-belleğin) boyutuna dönüş yapar.

```
int  StringBufferLen(
   string  string_var      // dizgi
   )
```

Parametreler

string_var

[in] Dizgi.

Dönüş değeri

0 değeri dizginin sabit oluğunu ve tampon boyutunun değiştirilemeyeceğini ifade eder. -1 değeri ise dizginin müşteri terminaline ait olduğunu ifade eder, bu durumda tamponu değiştirmenin olası sonuçları vardır.

Örnek:

```
void OnStart()
  {
   long length=1000;
   string a="a",b="b";
//---
   long i;
   Print("önce: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = ",StringLen(a));
   for(i=0;i<length;i++)
     {
      StringAdd(a,b);
     }
   Print("sonra: StringBufferLen(a) = ",StringBufferLen(a),
         "  StringLen(a) = ",StringLen(a));
  }
```

Ayrıca Bakınız

StringAdd, StringInit, StringLen, StringFill

StringCompare

Bu fonksiyon iki dizgiyi karşılaştırır ve karşılaştırma sonucuna tam-sayı şeklinde dönüş yapar.

```
int  StringCompare(
   const string&  string1,                // ilk dizgi
   const string&  string2,                // ikinci dizgi
   bool           case_sensitive=true     // karşılaştırma için, durum hassasiyet mo
   );
```

Parametreler

string1

[in] Birinci dizgi.

string2

[in] İkinci dizgi.

case_sensitive=true

[in] Durum hassasiyeti modu. 'true' ise, "A">"a". 'false' ise, "A"="a". 'true' varsayılan değerdir.

Dönüş değeri

- string1<string2 ise, -1 (eksi bir)
- string1=string2 ise, 0 (sıfır)
- string1>string2 ise, 1 (bir)

Not

Dizgiler mevcut kod sayfasına göre alfabetik olarak, sembol sembol karşılaştırılır.

Örnek:

```
void OnStart()
  {
//--- hangisi daha büyük - apple veya home?
   string s1="Apple";
   string s2="home";

//--- duruma duyarlı karşılaştırma
   int result1=StringCompare(s1,s2);
   if(result1>0) PrintFormat("Duruma duyarlı karşılaştırma: %s > %s",s1,s2);
   else
     {
      if(result1<0)PrintFormat("Duruma duyarlı karşılaştırma: %s < %s",s1,s2);
      else PrintFormat("Duruma duyarlı karşılaştırma: %s = %s",s1,s2);
     }

//--- durumdan bağımsız karşılaştırma
   int result2=StringCompare(s1,s2,false);
   if(result2>0) PrintFormat("Durumdan bağımsız karşılaştırma: %s > %s",s1,s2);
   else
     {
```

```
      if(result2<0)PrintFormat("Durumdan bağımsız karşılaştırma: %s < %s",s1,s2);
      else PrintFormat("Durumdan bağımsız karşılaştırma: %s = %s",s1,s2);
     }
/* Sonuç:
    Duruma duyarlı karşılaştırma: Apple < home
    Durumdan bağımsız karşılaştırma: Apple < home
*/
  }
```

Ayrıca Bakınız

String Tipi, CharToString(), ShortToString(), StringToCharArray(), StringToShortArray(), StringGetCharacter(), Kod Sayfasının Kullanımı

StringConcatenate

Geçirilen parametrelerden bir dizgi şekillendirir ve şekillendirilen dizginin büyüklüğüne dönüş yapar. Parametreler herhangi bir tipte olabilirler. Parametreler 2'den az veya 64'ten fazla olamaz.

```
int  StringConcatenate(
   string&  string_var,    // şekillendirilecek dizgi
   void argument1          // herhangi bir basit tipteki ilk parametre
   void argument2          // herhangi bir basit tipteki ikinci parametre
   ...                     // herhangi bir basit tipteki sonraki parametre
   );
```

Parametreler

string_var

[out] Birleştirme yoluyla şekillendirilecek dizgi.

argumentN

[in] Virgülle ayrılmış değerler. 2 ile 63 adet arasında basit tipli parametreler.

Dönüş değeri

Parametrelerin birleştirilmesi ve string tipine dönüştürülmesiyle elde edilen dizginin uzunluğu. Parametreler, Print() ve Comment() fonksiyonlarında kullanılan aynı kurallarla dönüştürülürler.

Ayrıca Bakınız

StringAdd, StringSplit, StringSubstr

StringFill

Seçilen bir dizgiyi belirtilen sembollerle doldurur.

```
bool  StringFill(
   string&   string_var,        // doldurulacak dizgi
   ushort    character          // dizgiyi dolduracak sembol
   );
```

Parametreler

string_var

[in][out] Seçilen sembolle doldurulacak dizgi.

character

[in] Dizginin doldurulacağı sembol.

Dönüş değeri

Başarı durumunda 'true', aksi durumda 'false' dönüşü yapar. Hata kodunu alabilmek için GetLastError() fonksiyonunu çağırın.

Not

Dizgi, yerinde doldurulur; yani, yeni dizgi oluşturulması veya kopyalama gibi geçiş işlemleri olmadan, semboller doğrudan dizgi içine yerleştirilir. Bu, işlem zamanından kazanılmasını sağlar.

Örnek:

```
void OnStart()
  {
   string str;
   StringInit(str,20,'_');
   Print("str = ",str);
   StringFill(str,0);
   Print("str = ",str,": StringBufferLen(str) = ", StringBufferLen(str));
  }
// Sonuç
//   str = ____________________
//   str =  : StringBufferLen(str) = 20
//
```

Ayrıca Bakınız

StringBufferLen, StringLen, StringInit

StringFind

Dizgi içinde bir alt-dizgiyi arar.

```
int  StringFind(
   string  string_value,        // aramanın yapılacağı dizgi
   string  match_substring,     // aranan
   int     start_pos=0          // arama hangi konumdan başlayacak
   );
```

Parametreler

string_value

[in] Aramanın yapıldığı dizgi.

match_substring

[in] Aranan alt-dizgi.

start_pos=0

[in] Dizgi içinde aramanın başlayacağı konum.

Dönüş değeri

Alt-dizginin başladığı konuma dönüş yapar, alt-dizgi bulunamazsa -1 dönüşü yapar.

Ayrıca Bakınız

StringSubstr, StringGetCharacter, StringLen, StringLen

StringGetCharacter

Dizginin belirli bir konumundaki sembolün değerine dönüş yapar.

```
ushort  StringGetCharacter(
   string  string_value,      // dizgi
   int     pos                // sembolün dizgi içindeki konumu
   );
```

Parametreler

string_value

[in] Dizgi.

pos

[in] Sembolün dizgi içindeki konumu. 0 ile StringLen(metin) -1, aralığında olabilir.

Dönüş değeri

Sembol kodu veya hata durumunda 0. Hata kodunu alabilmek için GetLastError() fonksiyonunu çağırın.

Ayrıca Bakınız

StringSetCharacter, StringBufferLen, StringLen, StringFill, StringInit, StringToCharArray, StringToShortArray

StringInit

Belirtilen sembollerle bir dizgiyi başlatır ve belirtilen dizgi uzunluğunu verir.

```
bool  StringInit(
   string&    string_var,       // başlatılacak dizgi
   int        new_len=0,        // başlatmadan sonraki (istenen) dizgi uzunluğu
   ushort     character=0       // dizginin doldurulacağı sembol.
   );
```

Parametreler

string_var

[in][out] Başlatılacak veya sonlandırılacak dizgi.

new_len=0

[in] Başlatmadan sonraki dizgi uzunluğu. length=0 ise, dizgi sonlandırılır, yani dizgi tamponu (arabelleği) boşaltılır ve tampon adresi sıfırlanır.

character=0

[in] Dizgiyi doldurmak için kullanılacak sembol.

Dönüş değeri

Başarı durumunda 'true', aksi durumda 'false' dönüşü yapar. Hata kodunu alabilmek için GetLastError() fonksiyonunu çağırın.

Not

character=0 ve new_len>0 ise, istenen uzunlukta dizgi için tampon tahsis edilir ve sıfırlarla doldurulur. Dizgi uzunluğu sıfıra eşit olacaktır, çünkü tüm tampon, dizgi sonlandırıcılarla doldurulmuş olacaktır.

Örnek:

```
void OnStart()
  {
//---
   string str;
   StringInit(str,200,0);
   Print("str = ",str,": StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
  }
/* Sonuç
str = : StringBufferLen(str) = 200   StringLen(str) = 0
*/
```

Ayrıca Bakınız

StringBufferLen, StringLen

StringLen

Bir dizgideki sembollerin sayısına dönüş yapar.

```
int  StringLen(
   string  string_value      // dizgi
   );
```

Parametreler

string_value

[in] Uzunluğu hesaplanacak dizgi.

Dönüş değeri

Dizgideki sembollerin sayısı (sonlandırma sıfırı olmadan).

Ayrıca Bakınız

StringBufferLen, StringTrimLeft, StringTrimRight, StringToCharArray, StringToShortArray

StringSetLength

Bir dizge için belirtilen bir uzunluk (karakter olarak) ayarlar.

```
bool  StringSetLength(
   string&     string_var,       // dizge
   uint        new_length        // yeni dizge uzunluğu
   );
```

Parametreler

string_var

[in][out] Karakter cinsinden yeni bir uzunluk belirlemek istediğiniz dizge.

new_capacity

[in] Karakter cinsinden gerekli dizge uzunluğu. Eğer new_length, geçerli boyuttan küçükse, fazla karakterler atılır.

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

StringSetLength() fonksiyonu bir dizge için ayrılan tampon boyutunu değiştirmez.

Ayrıca bakınız

StringLen, StringBufferLen, StringReserve StringInit, StringSetCharacter

StringReplace

Dizgi içinde bulunan belirli alt-dizgileri, belirtilen başka alt-dizgilerle değiştirir.

```
int  StringReplace(
   string&         str,              // içinde değişim yapmak istediğimiz dizgi
   const string    find,             // aranan alt-dizgi
   const string    replacement       // bulunan yere yerleştirilecek alt-dizgi
   );
```

Parametreler

str

[in][out] İçinde değişim yapmak istediğimiz dizgi.

find

[in] Yenisiyle değiştirilecek olan alt dizgi.

replacement

[in] Bulunanın yerine girilecek olan alt dizgi.

Dönüş değeri

Başarı durumunda değiştirilen alt-dizgilerin sayısına, aksi durumda -1 değerine dönüş yapar. Bir hata kodu almak için GetLastError() fonksiyonunu kullanın.

Not

Eğer fonksiyon başarılı olmuşsa ama herhangi bir değişim yapılmamışsa (değiştirilecek alt-dizgi bulunamamışsa), 0 dönüşü yapar.

Hata, yanılış *str* veya *find* parametrelerinden kaynaklanabilir (boş veya başlatılmamış dizgi, bakınız StringInit()). Ayrıca, değişimi yapmaya yetecek bellek olmaması durumunda da hata oluşur.

Örnek:

```
  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("Değiştirilen: ", replaced,". Sonuç=",text);

// Sonuç
//  Değiştirilen: 3. Sonuç=The slow black bear jumped over the lazy dog.
//
```

Ayrıca Bakınız

StringSetCharacter(), StringSubstr()

StringReserve

Bellekteki bir dizge için belirtilen boyutta bir tampon ayırır.

```
bool  StringReserve(
   string&     string_var,        // dizge
   uint        new_capacity       // Bir dizge depolamak için tampon boyutu
   );
```

Parametreler

string_var

[in][out] Tampon boyutunu değiştirmek istediğiniz dizge.

new_capacity

[in] Bir dizge için gerekli tampon boyutu. Eğer new_capacity boyutu dizge uzunluğundan küçükse, geçerli tamponun boyutu değişmez.

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

Genel olarak, dizge boyutu dizge depolamak için kullanılan tamponun boyutuna eşit değildir. Bir dizge oluştururken, uygun tampon genellikle bir kenar boşluğuyla ayrılır. StringReserve() fonksiyonu, tampon boyutunu yönetmeyi sağlar ve gelecekteki işlemler için en uygun boyutu belirtir.

StringInit() fonksiyonunun aksine, StringReserve() fonksiyonu dizge içeriğini değiştirmez ve bu içeriği karakterlerle doldurmaz.

Örnek:

```
void OnStart()
  {
   string s;
//--- StringReserve kullanmadan operasyon hızını kontrol et
   ulong t0=GetMicrosecondCount();
   for(int i=0; i< 1024; i++)
      s+=" "+(string)i;
   ulong msc_no_reserve=GetMicrosecondCount()-t0;
   s=NULL;
//--- şimdi, StringReserve kullanarak aynı şeyi yap
   StringReserve(s,1024 * 3);
   t0=GetMicrosecondCount();
   for(int i=0; i< 1024; i++)
      s+=" "+(string)i;
   ulong msc_reserve=GetMicrosecondCount()-t0;
//--- zamanı kontrol et
   Print("StringReserve ile test şu kadar sürdü: "+(string)msc_reserve+" msc");
   Print("StringReserve olmadan test şu kadar sürdü: "+(string)msc_no_reserve+" msc");
/* Sonuç:
```

```
    StringReserve ile test şu kadar sürdü: 50 msc
    StringReserve olmadan test şu kadar sürdü: 121 msc
*/
  }
```

Ayrıca bakınız

StringBufferLen, StringSetLength, StringInit, StringSetCharacter

StringSetCharacter

Belirtilen konumundaki sembol değeri değiştirilmiş bir dizginin kopyasına dönüş yapar.

```
bool  StringSetCharacter(
   string&    string_var,        // dizgi
   int        pos,               // konum
   ushort     character          // karakter
   );
```

Parametreler

string_var

[in][out] Dizgi.

pos

[in] Karakterin dizgi içindeki konumu. 0 ile StringLen(metin) aralığında olabilir.

character

[in] Sembol kodu (Unicode).

Dönüş değeri

Başarılı sonuç durumunda 'true', aksi durumda ise 'false' dönüşü yapar. Bir hata kodu almak için, GetLastError() fonksiyonunu çağırın.

Not

pos değeri dizgi uzunluğundan küçükse ve sembol kodu değeri 0 ise, dizgi budanır (ama dizgi için ayrılan tampon boyutu aynı kalır). Dizgi değeri 'pos' değerine eşit olur.

pos değeri, dizgi uzunluğuna eşitse, belirtilen sembol dizginin sonuna eklenir ve uzunluk bir artar.

Örnek:

```
void OnStart()
  {
   string str="0123456789";
   Print("önce: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- ortasına sıfır ekle
   StringSetCharacter(str,6,0);
   Print("sonra: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
//--- sonuna bir sembol ekle
   int size=StringLen(str);
   StringSetCharacter(str,size,'+');
   Print("ek: str = ",str,",StringBufferLen(str) = ",
         StringBufferLen(str),"  StringLen(str) = ",StringLen(str));
  }
/* Sonuç
   önce: str = 0123456789 ,StringBufferLen(str) = 0   StringLen(str) = 10
    sonra: str = 012345 ,StringBufferLen(str) = 16   StringLen(str) = 6
```

```
   ek: str = 012345+ ,StringBufferLen(str) = 16   StringLen(str) = 7
*/
```

Ayrıca Bakınız

StringBufferLen, StringLen, StringFill, StringInit, CharToString, ShortToString, CharArrayToString, ShortArrayToString

StringSplit

Belirtilen ayracı kullanarak, belirtilen bir dizgiden alt-dizgiler elde eder ve bu alt-dizgilerin sayısına dönüş yapar.

```
int  StringSplit(
   const string   string_value,       // aranacak dizgi
   const ushort   separator,          // alt-dizgileri ayırmak için kullanılan ayraç
   string         & result[]          // bulunan alt-dizgilerin alınması için referans
   );
```

Parametreler

string_value

[in] Alt dizginin içinde bulunması istenilen dizgi (dizgi değişmez).

pos

[in] Ayırıcı karakterin kodu. Kodu almak için StringGetCharacter() fonksiyonunu kullanabilirsiniz.

result[]

[out] Elde edilen alt-dizgilerin yerleştirildiği (dizgilerden oluşan) bir dizi.

Dönüş değeri

result[] dizisindeki alt-dizgilerin sayısı. Ayraç, geçirilen dizgi içerisinde bulunamazsa, sadece kaynak dizgisi dizi içine geçirilir.

string_value değeri NULL veya boş ise, fonksiyon sıfır dönüşü yapar. Hata durumunda ise -1 dönüşü yapar. Hata kodu almak için GetLastError() fonksiyonunu kullanın.

Örnek:

```
string to_split="_life_is_good_"; // Alt-dizgilere ayrılacak dizgi
   string sep="_";                // Karakter şeklinde bir ayraç
   ushort u_sep;                  // Ayırıcı karakterin kodu
   string result[];               // Dizgilerin geçirileceği dizi
   //--- Ayraç kodunu al
   u_sep=StringGetCharacter(sep,0);
   //--- Dizgiyi alt-dizgilere böl
   int k=StringSplit(to_split,u_sep,result);
   //--- Bir yorum göster
   PrintFormat("Alınan dizgi: %d. Kullanılan ayraç: '%s', ayraç kodu: %d",k,sep,u_sep)
   //--- Şimdi, alınan tüm dizgileri çıktıla
   if(k>0)
     {
      for(int i=0;i<k;i++)
        {
         PrintFormat("result[%d]=\"%s\"",i,result[i]);
        }
     }
```

Ayrıca Bakınız

StringReplace(), StringSubstr(), StringConcatenate()

StringSubstr

Bir metnin belirtilen konumundan bir alt-dizgiyi ayıklar.

```
string  StringSubstr(
   string  string_value,      // dizgi
   int     start_pos,         // başlangıç konumu
   int     length=-1          // ayıklanacak dizginin uzunluğu
   );
```

Parametreler

string_value

[in] Alt-dizginin ayıklanacağı ana dizgi.

start_pos

[in] Alt-dizginin başlangıç konumu. 0 ile StringLen(metin) -1, aralığında olabilir.

length=-1

[in] Ayıklanacak alt-dizginin uzunluğu. Parametre değeri ayarlanmamışsa veya -1 değerine eşitse, başlangıç konumundan dizginin sonuna kadar ayıklama yapılır.

Dönüş değeri

Mümkünse, ayıklanan alt-dizginin bir kopyası. Aksi durumda boş bir dizgi.

Ayrıca Bakınız

StringSplit, StringFind, StringGetCharacter

StringToLower

Bir dizginin tüm sembollerini, konuma göre küçük karaktere çevirir.

```
bool  StringToLower(
   string&  string_var      // işlenecek dizgi
   );
```

Parametreler

string_var

[in][out] Dizgi.

Dönüş değeri

Başarı durumunda 'true', aksi durumda 'false' dönüşü yapar. Hata kodunu alabilmek için GetLastError() fonksiyonunu çağırın.

Ayrıca Bakınız

StringToUpper, StringTrimLeft, StringTrimRight

StringToUpper

Bir dizginin tüm sembollerini, konuma göre büyük karakterlere çevirir.

```
bool  StringToUpper(
   string&  string_var      // işlenecek dizgi
   );
```

Parametreler

string_var

[in][out] Dizgi.

Dönüş değeri

Başarı durumunda 'true', aksi durumda 'false' dönüşü yapar. Hata kodunu alabilmek için GetLastError() fonksiyonunu çağırın.

Ayrıca Bakınız

StringToLower, StringTrimLeft, StringTrimRight

StringTrimLeft

Dizginin sol tarafındaki ilk anlamlı sembolden önce gelen satır atlama karakterlerini, boşlukları ve sekmeleri keser. Dizgi, yerinde şekillendirilir.

```
int  StringTrimLeft(
   string&  string_var      // kırpılacak dizgi
   );
```

Parametreler

string_var

[in][out] Soldan kırpılacak dizgi.

Dönüş değeri

Kırpılan sembollerin sayısına dönüş yapar.

Ayrıca Bakınız

StringTrimRight, StringToLower, StringToUpper

StringTrimRight

Dizginin sağ tarafındaki son anlamlı sembolün ardından gelen satır atlama karakterlerini, boşlukları ve sekmeleri keser. Dizgi, yerinde şekillendirilir.

```
int  StringTrimRight(
   string&  string_var      // kırpılacak dizgi
   );
```

Parametreler

string_var

[in][out] Sağdan kırpılacak dizgi.

Dönüş değeri

Kırpılan sembollerin sayısına dönüş yapar.

Ayrıca Bakınız

StringTrimLeft, StringToLower, StringToUpper

Tarih ve Zaman

Bu, datetime tipi (1 Ocak 1970 tarihinin 0 anından beri geçen saniyeler şeklinde bir sayıyı temsil eden tamsayı tipi) verilerle çalışmak için tasarlanmış bir fonksiyonlar grubudur.

Yüksek kesinlikli sayaçlar düzenleyebilmek için, milisaniye bazında değerler sağlayan GetTickCount() fonksiyonunu kullanın.

Fonksiyon	Eylem
TimeCurrent	Bilinen son sunucu zamanına (son kotasyonun alındığı zamana) datetime biçiminde dönüş yapar
TimeTradeServer	Alım-satım sunucusunun mevcut hesaplanmış zamanına dönüş yapar
TimeLocal	Yerel bilgisayar zamanına datetime biçiminde dönüş yapar
TimeGMT	Müşteri terminalinin çalışmakta olduğu bilgisayarın yerel zamanını kullanarak, Yaz Saati Uygulaması ile, GMT değerine datetime biçiminde dönüş yapar
TimeDaylightSavings	Yaz Saati Uygulamasının değişim işaretine dönüş yapar
TimeGMTOffset	GMT zamanıyla yerel bilgisayar zamanı arasındaki farka, saniye bazında ve Yaz Saati Uygulamasını göz önünde bulundurarak dönüş yapar
TimeToStruct	Bir datetime değişkeninin değerini MqlDateTime yapı tipine dönüştürür
StructToTime	MqlDateTime yapı tipindeki bir değişkenin tipini datetime tipine dönüştürür

TimeCurrent

"Veri Penceresinden" seçilen belirli bir sembolde alınmış son kotasyonun anına, yani bilinen son sunucu zamanına dönüş yapar. Bu fonksiyon, OnTick() işleyicisi içinde alınmış ve işlenmiş son tikin zamanına dönüş yapar. Diğer durumlarda ise (örneğin, OnInit(), OnDeinit(), OnTimer() vb. işleyicilerin çağrılarında), "Piyasa Gözlemi" penceresinde yer alan her sembol için, alınan son kotasyonun zamanına dönüş yapar, zaman yine bu pencerenin başlığında görüntülenir. Zaman değeri sunucuda şekillenir ve bilgisayarınızdaki zamana bağlı değildir. Fonksiyonun iki versiyonu bulunur.

Parametresiz çağrı

```
datetime  TimeCurrent();
```

MqlDateTime tipi parametre ile çağrı

```
datetime  TimeCurrent(
   MqlDateTime&  dt_struct      // yapı tipli değişken
   );
```

Parametreler

dt_struct

[out] MqlDateTime yapı tipli değişken.

Dönüş değeri

datetime tipli değer

Not

MqlDateTime yapı tipli değişken, parametre olarak geçirilmişse, uygun şekilde doldurulur.

Yüksek kesinlikte sayaçlar ve zamanlayıcılar için, milisaniye bazında değerler üreten GetTickCount() fonksiyonunu kullanın.

During testing in the strategy tester, TimeCurrent() is simulated according to historical data.

TimeTradeServer

Alım-satım sunucusunun hesaplanan mevcut zamanına dönüş yapar. TimeCurrent() fonksiyonunun aksine, zaman değerinin hesaplanması müşteri terminalinde yapılır ve bilgisayarınızın zamanına bağlıdır. Fonksiyonun iki çeşidi vardır.

Parametresiz çağrı

```
datetime  TimeTradeServer();
```

MqlDateTime tipi parametre ile çağrı

```
datetime  TimeTradeServer(
   MqlDateTime&  dt_struct      // Yapı tipli değişken
   );
```

Parametreler

dt_struct

[out] MqlDateTime yapı tipli değişken .

Dönüş değeri

datetime tipli değer

Not

MqlDateTime yapı tipli değişken, parametre olarak geçirilmişse, uygun şekilde doldurulur.

Yüksek kesinlikte sayaçlar ve zamanlayıcılar için, milisaniye bazında değerler üreten GetTickCount() fonksiyonunu kullanın.

During testing in the strategy tester, TimeTradeServer() is always equal to TimeCurrent() simulated server time.

TimeLocal

Müşteri terminalinin çalıştırıldığı bilgisayarın yerel zamanına dönüş yapar. Fonksiyonun 2 çeşidi bulunmaktadır.

Parametresiz çağrı

```
datetime  TimeLocal();
```

MqlDateTime tipi parametre ile çağrı

```
datetime  TimeLocal(
   MqlDateTime&  dt_struct      // Yapı tipli değişken
   );
```

Parametreler

dt_struct

[out] MqlDateTime yapı tipli değişken .

Dönüş değeri

datetime tipli değer

Not

MqlDateTime yapı tipli değişken, parametre olarak geçirilmişse, uygun şekilde doldurulur.

Yüksek kesinlikte sayaçlar ve zamanlayıcılar için, milisaniye bazında değerler üreten GetTickCount() fonksiyonunu kullanın.

During testing in the strategy tester, TimeLocal() is always equal to TimeCurrent() simulated server time.

TimeGMT

GMT zamanına dönüş yapar. Yaz Saati dönüşümü hesaba katılarak, müşteri terminalinin çalıştırıldığı bilgisayarın zamanını kullanarak hesaplanır. Fonksiyonun 2 çeşidi bulunmaktadır.

Parametresiz çağrı

```
datetime  TimeGMT();
```

MqlDateTime tipi parametre ile çağrı

```
datetime  TimeGMT(
   MqlDateTime&  dt_struct      // Yapı tipli değişken
   );
```

Parametreler

dt_struct

[out] MqlDateTime yapı tipli değişken .

Dönüş değeri

datetime tipli değer

Not

MqlDateTime yapı tipli değişken, parametre olarak geçirilmişse, uygun şekilde doldurulur.

Yüksek kesinlikte sayaçlar ve zamanlayıcılar için, milisaniye bazında değerler üreten GetTickCount() fonksiyonunu kullanın.

During testing in the strategy tester, TimeGMT() is always equal to TimeTradeServer() simulated server time.

TimeDaylightSavings

Yaz saatine geçildiğinde, Yaz Saati Uygulaması için düzeltme değerine saniye bazında dönüş yapar. Bilgisayarınızın zaman ayarlarına bağlıdır.

```
int  TimeDaylightSavings();
```

Dönüş değeri

Kış zamanına (standart zamana) geçildiğinde, 0 dönüşü yapar.

TimeGMTOffset

Yaz saatini hesaba katarak, GMT zamanı ile yerel bilgisayar zamanı arasındaki saniye bazındaki farka dönüş yapar. Bilgisayarınızdaki zaman ayarlarına bağlıdır.

```
int  TimeGMTOffset();
```

Dönüş değeri

Yerel bilgisayar zamanı ile GMT zamanı arasındaki mevcut farkı saniye bazında ifade eden, int tipli değer.

```
TimeGMTOffset() =  TimeGMT() - TimeLocal()
```

TimeToStruct

datetime tipi (01.01.1970 tarihinden bu yana geçen saniyeler) bir değeri, MqlDateTime tipi bir yapı değişkenine dönüştürür.

```
bool  TimeToStruct(
   datetime      dt,            // tarih ve zaman
   MqlDateTime&  dt_struct      // değerlerin alınması için bir yapı
   );
```

Parametreler

dt

[in] Dönüştürülecek tarih değeri.

dt_struct

[out] MqlDateTime yapı tipli değişken .

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

StructToTime

MqlDateTime yapısındaki bir değişkeni datetime tipli bir değere dönüştürür ve sonuç değerine dönüş yapar .

```
datetime  StructToTime(
   MqlDateTime&  dt_struct      // tarih ve zaman yapısı
   );
```

Parametreler

dt_struct

[in] MqlDateTime yapı tipli bir değişken.

Dönüş değeri

01.01.1970'den beri geçen saniyeler şeklinde datetime tipi bir değer.

Hesap Bilgisi

Mevcut hesabın parametrelerini veren fonksiyonlar

Fonksiyon	Eylem
AccountInfoDouble	Karşılık gelen hesap özelliğinin double tipli değerine dönüş yapar
AccountInfoInteger	Karşılık gelen hesap özelliğinin tamsayı tipli değerine dönüş yapar
AccountInfoString	Karşılık gelen hesap özelliğinin string tipli değerine dönüş yapar

AccountInfoDouble

Karşılık gelen hesap özelliğinin değerine dönüş yapar.

```
double  AccountInfoDouble(
   ENUM_ACCOUNT_INFO_DOUBLE  property_id      // özellik tanımlayıcı
   );
```

Parametreler

property_id

[in] Özellik tanımlayıcı. ENUM_ACCOUNT_INFO_DOUBLE sayımının değerlerinden birini alabilir.

Dönüş değeri

double tipli değer.

Örnek:

```
void OnStart()
  {
//--- AccountInfoDouble() fonksiyonundan elde edilen mevcut bilgileri göster
   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));
  }
```

Ayrıca Bakınız

SymbolInfoDouble, SymbolInfoString, SymbolInfoInteger, PrintFormat

AccountInfoInteger

Hesap özelliklerinin değerine dönüş yapar.

```
long  AccountInfoInteger(
   ENUM_ACCOUNT_INFO_INTEGER  property_id      // Özellik tanımlayıcı
   );
```

Parametreler

property_id

[in] Özellik tanımlayıcı. Bu değer, ENUM_ACCOUNT_INFO_INTEGER sayımının değerlerinden biri olabilir.

Dönüş değeri

long tipli değer.

Not

Varlık, bool, int veya long tiplerinden biri olmalıdır.

Örnek:

```
void OnStart()
  {
//--- AccountInfoInteger() fonksiyonundan elde edilen mevcut bilgileri göster
   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

//--- Alım-satım yapılabilirliği hakkında bilgi ver
   if(thisAccountTradeAllowed)
      Print("Bu hesapta alım-satıma izin verilmiştir ");
   else
      Print("Bu hesapta alım-satım yasaktır!");

//--- Bu hesapta Uzman Danışman ile alım-satım yapılabilir mi öğren
   if(EATradeAllowed)
      Print("Bu hesapta Uzman Danışman ile alım-satıma izin verilmiştir");
   else
      Print("Bu hesapta Uzman Danışman İle alım-satım yasaklanmıştır!");

//--- Hesap tipini öğren
   switch(tradeMode)
     {
      case(ACCOUNT_TRADE_MODE_DEMO):
         Print("Bu bir deneme hesabı");
         break;
```

```
      case(ACCOUNT_TRADE_MODE_CONTEST):
         Print("Bu bir yarışma hesabı");
         break;
      default:Print("Bu bir gerçek hesap!");
     }

//--- StopOut seviyesi ayarlama modunu öğren
   switch(stopOutMode)
     {
      case(ACCOUNT_STOPOUT_MODE_PERCENT):
         Print("StopOut seviyesi yüzde olarak belirlenmiş");
         break;
      default:Print("StopOut seviyesi parasal terimlerle belirlenmiş");
     }
  }
```

Ayrıca Bakınız

Hesap Bilgisi

AccountInfoString

Karşılık gelen hesap özelliğinin değerine dönüş yapar.

```
string  AccountInfoString(
   ENUM_ACCOUNT_INFO_STRING  property_id      // Özellik tanımlayıcı
   );
```

Parametreler

property_id

[in] Özellik tanımlayıcı. Bu değer ENUM_ACCOUNT_INFO_STRING'in değerlerinden biri olabilir.

Dönüş değeri

string tipli değer.

Örnek:

```
void OnStart()
  {
//--- AccountInfoString() fonksiyonundan elde edilen mevcut bilgileri göster
   Print("Aracı kurumun adı = ",AccountInfoString(ACCOUNT_COMPANY));
   Print("Teminat para birimi= ",AccountInfoString(ACCOUNT_CURRENCY));
   Print("Müşteri adı = ",AccountInfoString(ACCOUNT_NAME));
   Print("Alım-satım sunucusunun adı = ",AccountInfoString(ACCOUNT_SERVER));
  }
```

Ayrıca Bakınız

Hesap Bilgisi

Durum Kontrolü

Müşteri terminalinin mevcut durumunun parametrelerine dönüş yapan fonksiyonlar

Fonksiyon	Eylem
GetLastError	Son hataya dönüş yapar
IsStopped	Bir MQL5 programı işlemi durdurma emri almışsa, 'true' değerine dönüş yapar
UninitializeReason	Sonlandırma sebebi koduna dönüş yapar
TerminalInfoInteger	Programın ortamının karşılık gelen özelliğinin tamsayı değerine dönüş yapar
TerminalInfoDouble	Programın ortamının karşılık gelen özelliğinin double tipli değerine dönüş yapar
TerminalInfoString	Programın ortamının karşılık gelen özelliğinin string tipli değerine dönüş yapar
MQLInfoInteger	Çalışmakta olan bir MQL5 programının karşılık gelen özelliğinin tamsayı değerine dönüş yapar
MQLInfoString	Çalışmakta olan bir MQL5 programının karşılık gelen özelliğinin string tipli değerine dönüş yapar
Symbol	Mevcut çizelgenin sembol ismine dönüş yapar
Period	Mevcut çizelge zaman aralığına dönüş yapar
Digits	Mevcut çizelge sembolünün fiyat değeri çözünürlüğünü belirleyen ondalık basamak sayısına dönüş yapar
Point	Mevcut sembolün karşıt dövizindeki puan büyüklüğüne dönüş yapar

GetLastError

_LastError sistem değişkeninin içeriğine dönüş yapar.

```
int  GetLastError();
```

Dönüş değeri

MQL5 programının çalıştırılması sırasında oluşturulan son error değerine dönüş yapar.

Not

Fonksiyon çağrısından sonra _LastError içeriği sıfırlanmaz. Bu değişkeni sıfırlamak için ResetLastError() fonksiyonunu çağırmalısınız.

Ayrıca Bakınız

Alım-Satım Sunucusunun Dönüş Kodları

IsStopped

Bir MQL5 programının zorla kapanmış olmasını kontrol eder.

```
bool  IsStopped();
```

Dönüş değeri

Eğer _StopFlag sistem değişkeni 0 harici bir değer içeriyorsa, 'true' değerine dönüş yapar. Eğer MQL5 programı, işlemini tamamlama komutu almışsa _StopFlag içine sıfır olmayan bir değer yazılır. Bu durumda programı hemen sonlandırmalısınız, aksi durumda 3 saniye sonra program dışarıdan tamamlanmaya zorlanacaktır.

UninitializeReason

Bir sonlandırma sebebi koduna dönüş yapar.

```
int  UninitializeReason();
```

Dönüş değeri

OnDeinit() fonksiyonunun çağrısından önce şekillenen _UninitReason değerine dönüş yapar. Değer, sonlandırmaya yol açan sebeplere bağlıdır.

TerminalInfoInteger

MQL5 programının karşılık gelen özelliğinin değerine dönüş yapar.

```
int  TerminalInfoInteger(
   int  property_id      // özellik tanımlayıcısı
   );
```

Parametreler

property_id

[in] Özelliğin tanımlayıcısı. ENUM_TERMINAL_INFO_INTEGER sayımının değerlerinden biri olabilir.

Dönüş değeri

int tipli değer.

TerminalInfoDouble

Mql5 program ortamının karşılık gelen özelliğinin değerini alır.

```
double  TerminalInfoDouble(
   int  property_id      // özellik tanımlayıcısı
   );
```

Parametreler

property_id

[in] Özellik tanımlayıcısı. Can be one of the values of the enumeration ENUM_TERMINAL_INFO_DOUBLE.

Dönüş Değeri

double tipli değer.

TerminalInfoString

Fonksiyon, MQL5 programının karşılık gelen özelliğinin değerine dönüş yapar. Özelliğin tipi string olmalıdır.

```
string  TerminalInfoString(
   int  property_id      // özellik tanımlayıcısı
   );
```

Parametreler

property_id

[in] Özelliğin tanımlayıcısı. ENUM_TERMINAL_INFO_STRING sayımının değerlerinden biri olabilir.

Dönüş değeri

string tipli değer.

MQLInfoInteger

Çalışan bir MQL5 programının karşılık gelen özelliğinin değerine dönüş yapar.

```
int  MQLInfoInteger(
   int  property_id      // özellik tanımlayıcısı
   );
```

Parametreler

property_id

[in] Özelliğin tanımlayıcısı. ENUM_MQL_INFO_INTEGER sayımının değerlerinden biri olabilir.

Dönüş değeri

int tipli değer.

MQLInfoString

Çalışan bir MQL5 programının karşılık gelen özelliğinin değerine dönüş yapar.

```
string  MQLInfoString(
   int  property_id      // Özellik tanımlayıcısı
   );
```

Parametreler

property_id

[in] Özelliğin tanımlayıcısı. ENUM_MQL_INFO_STRING sayımının değerlerinden biri olabilir.

Dönüş değeri

string tipli değer.

Symbol

Mevcut çizelgenin sembol ismine dönüş yapar.

```
string  Symbol();
```

Dönüş değeri

Mevcut çizelgenin sembol ismini muhafaza eden _Symbol sistem değişkeninin değerine dönüş yapar.

Dönüş değeri

Uzman Danışmanlar, göstergeler ve komut dosyalarından farklı olarak, hizmetler belirli bir grafiğe bağlı değildir. Bu nedenle, Symbol() bir hizmet için boş bir dizgi ("") geri döndürür.

Period

Mevcut çizelge zaman aralığına dönüş yapar.

```
ENUM_TIMEFRAMES  Period();
```

Dönüş değeri

Mevcut çizelge zaman aralığını içeren _Period değişkenine dönüş yapar. Değer, ENUM_TIMEFRAMES sayımının değerlerinden biri olabilir.

Dönüş değeri

Uzman Danışmanlar, göstergeler ve komut dosyalarından farklı olarak, hizmetler belirli bir grafiğe bağlı değildir. Bu nedenle, Period() bir hizmet için 0 geri döndürür.

Ayrıca Bakınız

PeriodSeconds, Çizelge zaman aralıkları, Zaman ve Tarih, Nesnelerin Görünürlüğü

Digits

Mevcut çizelge sembolünün fiyat değeri çözünürlüğünü belirleyen ondalık basamak sayısına dönüş yapar.

```
int  Digits();
```

Dönüş değeri

Mevcut çizelge sembolünün fiyat değeri çözünürlüğünü belirleyen ondalık basamak sayısını kaydeden _Digits değişkeninin değeri.

Point

Mevcut sembolün karşıt döviz biriminin puan büyüklüğüne dönüş yapar.

```
double  Point();
```

Dönüş değeri

Mevcut sembolün karşıt döviz biriminin puan büyüklüğünü saklayan _Point değişkeninin değerine dönüş yapar.

Olay yönetimi

MQL5 dili önceden tanımlanmış belirli olayların yönetilmesini sağlar. Bu olayları yönetmek için, bir MQL5 programında fonksiyonların tanımlanmış olması gerekmektedir: fonksiyon adı, dönüş tipi, (varsa) bir parametre kümesi ve tipleri, olay yönetimi fonksiyonunun açıklamasına tam anlamıyla uyumlu olmalıdır.

Müşteri terminali olay yöneticisi, bir olayı yöneten fonksiyonları tanımlamak için dönüş ve parametre tiplerini kullanır. Eğer belirli bir fonksiyon aşağıdaki açıklamalara uyumlu olmayan bir dönüş tipine ve bazı parametrelere sahipse, bu şekildeki bir fonksiyon bir olayı yönetmek için kullanılamaz.

Fonksiyon	Açıklama
OnStart	Fonksiyon, Start olayının skriptte ayarlanan bazı eylemleri gerçekleştirmek için oluştuğunda çağrılır.
OnInit	Fonksiyon, çalışan bir MQL5 programını başlatmak için Init olayı oluştuğunda göstergeler ve uzman danışmanlarda çağrılır.
OnDeinit	Fonksiyon, çalışan bir MQL5 programını sonlandırmak için Deinit olayı oluştuğunda göstergeler ve uzman danışmanlarda çağrılır.
OnTick	Fonksiyon, yeni bir fiyatı(tik) işlemek için NewTick olayı oluştuğunda uzman danışmanlarda çağrılır.
OnCalculate	Fonksiyon, fiyat bilgisinin değişimini işlemek için Calculate olayı oluştuğunda göstergelerde çağrılır.
OnTimer	Fonksiyon, Timer periyodik olayının terminal tarafından sabit zaman aralıklarında oluşturulmasıyla göstergeler ve uzman danışmanlarda çağrılır.
OnTrade	Fonksiyon, bir işlem sunucusu üzerinde bir alım-satım işleminin sonunda oluşturulan Trade olayı sırasında uzman danışmanlarda çağrılır.
OnTradeTransaction	Fonksiyon, bir işlem isteği gerçekleştirim sonuçlarını işlemek için TradeTransaction olayı oluştuğunda uzman danışmanlarda çağrılır.
OnBookEvent	Fonksiyon, piyasa derinliğindeki değişiklikleri işlemek için BookEvent olayı oluştuğunda uzman danışmanlarda çağrılır.
OnChartEvent	Fonksiyon, bir MQL5 programı veya bir kullanıcı tarafından yapılan grafik değişikliklerini işlemek için ChartEvent olayı oluştuğunda göstergelerde ve uzman danışmanlarda çağrılır.
OnTester	Fonksiyon, bir uzman danışmanın geçmiş veriler üzerinde test edilmesinden sonra gerekli eylemleri gerçekleştirmek için Tester olayı oluştuğunda uzman danışmanlarda çağrılır.
OnTesterInit	Fonksiyon, strateji sınayıcısında optimizasyondan önce gerekli eylemleri gerçekleştirmek için TesterInit olayı oluştuğunda uzman danışmanlarda çağrılır.

Fonksiyon	Açıklama
OnTesterDeinit	Fonksiyon, strateji sınayıcısında uzman danışman optimizasyonundan sonra TesterDeinit olayı oluştuğunda uzman danışmanlarda çağrılır.
OnTesterPass	Fonksiyon, strateji sınayıcısında uzman danışman optimizasyonu sırasında yeni bir veri çerçevesinin gelişini işlemek için TesterPass olayı oluştuğunda uzman danışmanlarda çağrılır.

Müşteri terminali gelen olayları ilişkili olan açık grafiklere gönderir. Ayrıca, olaylar grafikler tarafından (grafik olayları) yada MQL5 programları (özel olaylar) tarafından oluşturulabilir . Grafiksel objenin oluşumu/silinimi olaylarının meydana gelmesi CHART_EVENT_OBJECT_CREATE ve CHART_EVENT_OBJECT_DELETE grafik özelliklerin ayarlanması ile etkinleştirilebilir/devre dışı bırakılabilir. Her bir MQL5 uygulaması ve grafiği, yeni gelen tüm olayların yerleştiği kendisine ait olaylar sırasına(kuyruğuna) sahiptir.

Bir program sadece üzerinde çalışmakta olduğu grafikten olayları alır. Tüm olaylar alındıkları sıraya göre birbiri peşi sıra yönetilir. Eğer sıra zaten NewTick olayını içeriyorsa veya bu olay yönetilme aşamasındaysa, o zaman yeni NewTick olayı MQL5 uygulama sırasına eklenmez. Benzer şekilde, eğer ChartEvent zaten bir MQL5 programı sırası içerisinde veya bu gibi bir olay yönetiliyorsa, o zaman bu tip yeni bir olay sıraya yerleştirilmez. Zamanlayıcı(Timer) olayının yönetimi de aynı şekilde işlemden geçirilir – eğer Timer olayı zaten sırada veya yönetiliyorsa, yeni zamanlayıcı olayı sıraya konulmaz.

Olay sıraları sınırlı ancak yeterli bir boyuta sahiptir, bu nedenle sıra taşması doğru bir şekilde geliştirilmiş program için olası değildir. Sıra taştığında, yeni olaylar sıraya yerleştirilmeden atılır.

Olayları yönetmek için sonsuz döngüleri kullanmamanız şiddetle önerilir. Olası istisnalar tek bir Start olayını işleyen komut dosyalarıdır.

Kütüphaneler hiç bir olayı yönetmez.

OnStart

Fonksiyon, Start olayının meydana gelmesi ile çağrılır. Fonksiyon komut dosyasında yer alan eylemlerin tek seferde yürütülmesi için tasarlanmıştır. Fonksiyon için 2 seçenek vardır.

Sonucu geri döndüren versiyon

```
int  OnStart(void);
```

Geri Dönüş Değeri

Günlük sekmesinde görülen int türünün değeri.

"script script_name removed (result code N)" girişi, komut dosyası yürütmesi tamamlandıktan sonra terminal günlüğünde oluşturulur. Burada N, OnStart() fonksiyonu tarafından geri döndürülen değerdir.

"service service_name stopped (result code N)" girişi, hizmet yürütmesi tamamlandıktan sonra terminal günlüğünde oluşturulur. Burada N, OnStart() fonksiyonu tarafından geri döndürülen değerdir.

Yürütme sonucunu geri döndüren OnStart() çağrısı sadece komut dosyası veya hizmet yürütülmesine izin verdiği için değil, ayrıca program yürütme sonucunu analiz etmek için bir hata kodu veya başka yararlı veriler geri döndürdüğünden dolayı kullanım için önerilmektedir.

Sonuç geri dönüşü olmayanversiyon sadece eski kodlarla uyumluluk için bırakılmıştır. Kullanım için önerilmemektedir

```
void  OnStart(void);
```

Not

Komut dosyalarında ve hizmetlerde olayları yönetmek için tek işlev onStart() işlevidir. Bu programlara başka hiçbir olay gönderilmez. Dolayısıyla, Start olayı Uzman Danışmanlara ve özel göstergelere iletilmez.

Örnek komut dosyası:

```
//--- renkler ile çalışmak için makrolar
#define XRGB(r,g,b)     (0xFF000000|(uchar(r)<<16)|(uchar(g)<<8)|uchar(b))
#define GETRGB(clr)     ((clr)&0xFFFFFF)
//+------------------------------------------------------------------+
//| Komut dosyası başlama fonksiyonu                                 |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- düşen mum rengini ayarla
   Comment("Düşen mum rengini ayarla");
   ChartSetInteger(0,CHART_COLOR_CANDLE_BEAR,GetRandomColor());
   ChartRedraw(); // yeni bir fiyatı beklemeden grafiği hemen güncelle
   Sleep(1000);   // tüm değişiklikleri görmek için 1 saniye duraklayın
//--- yükselen mum rengini ayarla
   Comment("Yükselen mum rengini ayarla");
   ChartSetInteger(0,CHART_COLOR_CANDLE_BULL,GetRandomColor());
```

```
   ChartRedraw();
   Sleep(1000);
//--- arka plan rengini ayarla
   Comment("Arka plan rengini ayarla");
   ChartSetInteger(0,CHART_COLOR_BACKGROUND,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- Satış çizgisinin rengini ayarla
   Comment("Satış çizgisinin rengini ayarla");
   ChartSetInteger(0,CHART_COLOR_ASK,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- Alış çizgisinin rengini ayarla
   Comment("Alış çizgisinin rengini ayarla");
   ChartSetInteger(0,CHART_COLOR_BID,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- düşen barın ve düşen mumun çerçeve rengini ayarla
   Comment("Düşen barın ve düşen mumun çerçeve rengini ayarla");
   ChartSetInteger(0,CHART_COLOR_CHART_DOWN,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- çizgi gafiği ve Doji mumlarının rengini ayarla
   Comment("Çizgi gafiği ve Doji mumlarının rengini ayarla");
   ChartSetInteger(0,CHART_COLOR_CHART_LINE,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- yükselen barın ve yükselen mumun çerçeve rengini ayarla
   Comment("Yükselen barın ve yükselen mumun çerçeve rengini ayarla");
   ChartSetInteger(0,CHART_COLOR_CHART_UP,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- eksenlerin, ölçeklerin ve OHLC çizgilerinin rengini ayarla
   Comment("Eksenlerin, ölçeklerin ve OHLC çizgilerinin rengini ayarla");
   ChartSetInteger(0,CHART_COLOR_FOREGROUND,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- ızgara rengini ayarla
   Comment("Izgara rengini ayarla");
   ChartSetInteger(0,CHART_COLOR_GRID,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- Son fiyat rengini ayarla
   Comment("Son fiyat rengini ayarla");
   ChartSetInteger(0,CHART_COLOR_LAST,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- Karı al ve Zararı durdur işlem seviyelerinin rengini ayarla
   Comment("Karı al ve Zararı durdur işlem seviyelerinin rengini ayarla");
```

```
   ChartSetInteger(0,CHART_COLOR_STOP_LEVEL,GetRandomColor());
   ChartRedraw();
   Sleep(1000);
//--- hacimlerin ve piyasaya giriş seviyelerinin rengini ayarla
   Comment("Hacimlerin ve piyasaya giriş seviyelerinin rengini ayarla");
   ChartSetInteger(0,CHART_COLOR_VOLUME,GetRandomColor());
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| Rastgele oluşturulmuş bir rengi geri döndür                      |
//+------------------------------------------------------------------+
color GetRandomColor()
  {
   color clr=(color)GETRGB(XRGB(rand()%255,rand()%255,rand()%255));
   return clr;
  }
```

Ayrıca bakınız

Olay yönetimi fonksiyonları, Program yürütme, Müşteri terminali olayları

OnInit

Fonksiyon, Init olayı meydana geldiğinde göstergeler ve uzman danışmanlarda çağrılır. Çalışan bir MQL5 programını başlatmak için kullanılır. Fonksiyon için iki seçenek vardır.

Sonucu geri döndüren versiyon

```
int  OnInit(void);
```

Geri dönüş değeri

int tip değeri sıfır, başarılı başlatma anlamına gelmektedir.

Gerçekleşim sonucunu geri döndüren OnInit() çağrısı sadece program başlatmasına izin verdiği için değil, ayrıca programın erken sonlandırılması durumunda hata kodunu geri döndürdüğünden dolayı kullanım için önerilmektedir.

Sonuç geri dönüşü olmayanversiyon sadece eski kodlarla uyumluluk için bırakılmıştır. Kullanım için önerilmemektedir

```
void  OnInit(void);
```

Not

Init olayı, bir uzman danışmanı veya bir indikatörü yükledikten hemen sonra oluşturulur. Olay, komut dosyaları için oluşturulmaz. OnInit() fonksiyonu bir MQL5 programını başlatmak için kullanılır. Eğer OnInit() int tipi geri dönüş değerine sahipse, sıfırdan farklı geri dönüş kodu, başarısız başlatma anlamanına gelmektedir ve REASON_INITFAILED sonlandırma neden kodu ile Deinit olayını oluşturur.

Void tipinin OnInit() fonksiyonu her zaman başarılı başlatma anlamına gelmektedir ve kullanım için önerilmemektedir.

Uzman danışman girdilerini optimize etmek için ENUM_INIT_RETCODE listesindeki değerleri bir geri dönüş kodu olarak kullanılması önerilir. Bu değerler, en uygun test temsilcilerinın seçimi de dahil olmak üzere, optimizasyon işlemi yönetimini oragize etmeyi amaçlanmaktadır. Testi çalıştırmadan önce uzman danışman başlatması sırasında, TerminalInfoInteger() fonksiyonunu kullanarak temsilci yapısı ve kaynakları (çekirdek sayısı, serbest bellek miktarı vb.) hakkında verileri talep edebilirsiniz. Elde edilen verilere dayanarak, test temsilcisinin kullanılmasına izin verebilir veya onu uzman danışman optimizasyonunu yapmasından yasaklayabilirsiniz.

Tanımlayıcı	Açıklama
INIT_SUCCEEDED	Başlatma başarılı, uzman danışman testi devam ettirilebilir. Bu kod sıfır değeri ile aynı anlamdadır - test aygıtındaki uzman danışman başlatımı başarılıdır.
INIT_FAILED	Başlatma başarısızdır. Kaçınılmaz hatalardan dolayı teste devam etmenin bir anlamı yoktur. Örneğin, uzman danışman işleyişi için gerekli bir gösterge oluşturmak imkansızdır. Bu değerin geri dönüşü sıfırdan farklı bir değerin geri dönüşü ile aynı anlamdadır - test aygıtındaki uzman danışman başlatımı başarısızdır.
INIT_PARAMETERS_INCORRECT	Bir programlayıcı tarafından yanlış bir girdi parametre kümesini belirtmek için tasarlanmıştır. Genel optimizasyon tablosunda,

Tanımlayıcı	Açıklama
	bu geri dönüş kodlu sonuç dizisi kırmızı olarak vurgulanacaktır. Bu gibi bir uzman danışman girdiler kümesi için bir test gerçekleştirilmemiştir . Temsilci, yeni bir görev almak için hazırdır. Bu değer alındığında, strateji sınayıcısı bu görevi tekrar gerçekleşimi için diğer temsilcilere aktarmaz.
INIT_AGENT_NOT_SUITABLE	Başlatma sırasında herhangi bir program gerçekleşim hatası yoktur. Ancak, bazı nedenlerden dolayı, temsilci, testi yürütmek için uygun değildir. Örneğin, yeterli RAM yoktur, OpenCL desteği yoktur, vb. Bu kodun geri dönüşü ile, temsilci bu optimizasyonun sonuna kadar artık görev almayacaktır.

Strateji sınayıcısında INIT_FAILED/INIT_PARAMETERS_INCORRECT i geri döndüren OnInit() i uzman danışmanları optimize ederken kullanmanın dikkate alınması gereken bazı alışılmadık durumları vardır:

- OnInit() fonksiyonunun INIT_PARAMETERS_INCORRECT ı geri döndürmesini içeren parametrelerin kümesi, test için uygun kabul edilmez ve genetik optimizasyon sırasında bir sonraki popülasyonu elde etmek için kullanılmaz. Çok fazla "atılmış" parametre seti, en uygun uzman danışman parametrelerini ararken yanlış sonuçlara yol açabilir. Arama algoritması, optimizasyon kriteri fonksiyonunun düzgün olduğunu ve tüm girdi parametrelerinde boşluk olmadığını varsayar.
- Eğer OnInit() fonksiyonu INIT_FAILED değerini geri döndürürse, bu, bir sınamanın başlatılamayacağı ve uzman danışmanın temsilcinin belleğinden kaldırılacağı anlamına gelir. Uzman danışman, yeni bir parametre seti ile bir sonraki geçişi gerçekleştirmek için tekrar yüklenir. Sonraki optimizasyon geçişini başlatmak, TesterStop()'ı çağırmaktan çok daha fazla zaman alır.

Bir uzman danışman için örnek OnInit() fonksiyonu

```
//--- girdi parametreleri
input int      ma_period=20; // hareketli ortalama zaman aralığı

//--- uzman danışmanda kullanılan gösstegenin yönetimi
int indicator_handle;
//+------------------------------------------------------------------+
//| Uzman danışman başlatma fonksiyonu                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ma_period geçerliliğini konrol et
   if(ma_period<=0)
     {
      PrintFormat("Geçersiz ma_period girdi değeri: %d",ma_period);
      return (INIT_PARAMETERS_INCORRECT);
     }
//--- optimizasyon sırasında
   if(MQLInfoInteger(MQL_OPTIMIZATION))
```

```
      {
       //--- temsilci için uygun RAM miktarını kontrol et
       int available_memory_mb=TerminalInfoInteger(TERMINAL_MEMORY_TOTAL);
       if(available_memory_mb<2000)
         {
          PrintFormat("Test temsilcisi için yetersiz bellek miktarı: %d MB",
                      available_memory_mb);
          return (INIT_AGENT_NOT_SUITABLE);
         }
      }
//--- göstergeyi kontrol et
   indicator_handle=iCustom(_Symbol,_Period,"My_Indicator",ma_period);
   if(indicator_handle==INVALID_HANDLE)
     {
      PrintFormat("My_Indicator yönetiminin oluşumu başarısız oldu. Hata kodu %d",
                  GetLastError());
      return (INIT_FAILED);
     }
//--- Uzman danışman başlatma başarılı oldu
   return(INIT_SUCCEEDED);
  }
```

Ayrıca bakınız

OnDeinit, Olay yönetimi fonksiyonları, Program yürütme, Müşteri terminali olayları, Değişkenlerin başlatılması, Nesneleri oluşturma ve silme

OnDeinit

Fonksiyon, Deinit olayı meydana geldiğinde Uzman Danışmanlar ve göstergelerde çağrılır. Çalışan bir MQL5 programını sonlandırmak için kullanılır.

```
void  OnDeinit(
   const int  reason          // sonlandırma neden kodu
   );
```

Parametreler

reason

[in] Sonlandırma neden kodu.

Geri dönüş değeri

Geri dönüş değeri yok

Not

Deinit olayı aşağıdaki durumlarda uzman danışmanlar ve göstergeler için oluşturulur:

- MQL5 programının bağlı olduğu bir sembolün veya grafik zaman aralığının değişimi nedeniyle yeniden başlatma öncesinde;
- Girdilerin değişimi nedeniyle yeniden başlatma öncesinde;
- bir MQL5 programını kaldırmadan önce.

Reason parametresi aşağıdaki değerlere sahip olabilir:

Sabit	Değer	Açıklama
REASON_PROGRAM	0	Uzman danışman ExpertRemove() fonksiyonunu çağırarak çalışmayı durdurdu
REASON_REMOVE	1	Program grafikten kaldırıldı
REASON_RECOMPILE	2	Program tekrar derlendi
REASON_CHARTCHANGE	3	Bir sembol veya grafik zaman aralığı değişti
REASON_CHARTCLOSE	4	Grafik kapandı
REASON_PARAMETERS	5	Girdiler bir kullanıcı tarafından değiştirildi
REASON_ACCOUNT	6	Başka bir hesap etkinleştirildi veya hesap ayarlarındaki değişikliklerden dolayı alım-satım sunucusuna tekrar bağlanma gerçekleştirildi
REASON_TEMPLATE	7	Başka bir grafik şablonu uygulandı
REASON_INITFAILED	8	OnInit() yöneticisi sıfırdan farklı bir değer geri döndürdü
REASON_CLOSE	9	Terminal kapandı

Uzman danışman sonlandırma neden kodları, UninitializeReason() fonksiyonu veya önceden tanımlı _UninitReason değişkeninden de elde edilebilir.

Uzman danışman için örnek OnInit() ve OnDeinit() fonksiyonları

```
input int fake_parameter=3;      // kullanışsız parametre
//+------------------------------------------------------------------+
//| Uzman danışman başlatma fonksiyonu                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Programın derlendiği yapının numarasını elde edin
   Print(__FUNCTION__," Build #",__MQLBUILD__);
//--- Yeniden başlatma nedeni ayrıca OnInit()'de de elde edilebilir
   Print(__FUNCTION__," Sonlandırma neden kodu, uzman danışman yeniden başlatma sırası
//--- Bir sonlandırma neden kodunu elde etmenin birinci yolu
   Print(__FUNCTION__," _UninitReason = ",getUninitReasonText(_UninitReason));
//--- Bir sonlandırma neden kodunu elde etmenin ikinci yolu
   Print(__FUNCTION__," UninitializeReason() = ",getUninitReasonText(UninitializeReaso
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Uzman danışman sonlandırma fonksiyonu                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- Bir sonlandırma neden kodunu elde etmenin birinci yolu
   Print(__FUNCTION__," Deinitialization reason code = ",reason);
//--- Bir sonlandırma neden kodunu elde etmenin ikinci yolu
   Print(__FUNCTION__," _UninitReason = ",getUninitReasonText(_UninitReason));
//--- Bir sonlandırma neden kodunu elde etmenin üçüncü yolu
   Print(__FUNCTION__," UninitializeReason() = ",getUninitReasonText(UninitializeReaso
  }
//+------------------------------------------------------------------+
//| Sonlandırma neden kodunun metinsel açıklamasını geri döndürmek   |
//+------------------------------------------------------------------+
string getUninitReasonText(int reasonCode)
  {
   string text="";
//---
   switch(reasonCode)
     {
      case REASON_ACCOUNT:
         text="Hesap değişti";break;
      case REASON_CHARTCHANGE:
         text="Sembol yada zaman aralığı değişti";break;
      case REASON_CHARTCLOSE:
         text="Grafik kapandı";break;
      case REASON_PARAMETERS:
         text="Girde-parametresi değişti";break;
      case REASON_RECOMPILE:
```

```
         text="Program "+__FILE__+" yeniden derlendi";break;
      case REASON_REMOVE:
         text="Program "+__FILE__+" grafikten kaldırıldı";break;
      case REASON_TEMPLATE:
         text="Yeni şablon grafiğe uygulandı";break;
      default:text="Başka bir neden";
     }
//---
   return text;
  }
```

Ayrıca bakınız

OnInit, Olay yönetimi fonksiyonları, Program yürütme, Müşteri terminal olayları, Sonlandırma neden kodları, Değişkenlerin kapsamı ve ömrü, Nesneleri oluşturma ve silme

OnTick

Fonksiyon, yeni bir fiyatı yönetmek için NewTick olayı oluştuğunda uzman danışmanlarda çağrılır.

```
void  OnTick(void);
```

Geri dönüş değeri

Geri dönüş değeri yok

Not

NewTick olayı, uzman danışmanın bağlı olduğu grafiğin sembolü için yeni bir fiyat alındıktan sonra sadece uzman danışmanlar için üretilir. Bir NewTick olayı onlar için üretilmediğinden, bir özel gösterge veya komut dosyasında OnTick() fonksiyonu tanımlamanın bir anlamı yoktur.

Tick olayı sadece uzman danışmanlar için üretilir, ancak bu, uzman danışmanların illa OnTick() fonksiyonuna sahip olması gerektiği anlamına gelmez; çünkü uzman danışmanlar için NewTick olayına ek olarak ayrıca Timer, BookEvent ve ChatEvent olayları da oluşturulur.

Tüm olaylar alındıkları sıraya göre birbiri peşi sıra yönetilir. Eğer sıra zaten NewTick olayını içeriyorsa veya bu olay yönetilme aşamasındaysa, o zaman yeni NewTick olayı MQL5 uygulama sırasına eklenmez.

NewTick olayı otomatik alım-satımın etkin olup olmadığına bakılmaksızın oluşturulur (OtomatikAlımSatım/Otomatik İşlem butonu). Etkin olmayan otomatik alım-satım sadece, uzman danışmandan alım-satım isteğini göndermesini yasaklanması anlamına gelmektedir. Uzman danışman operasyonu durdurulmamıştır.

OtomatikAlımSatım butonuna basılarak otomatik alım-satımı devre dışı bırakmak, OnTick() fonksiyonunun geçerli gerçekleşimini kesintiye uğratmaz.

OnTick() fonksiyonundaki tüm alım-satım mantığını içeren **uzman danışman örneği**

```
//+------------------------------------------------------------------+
//|                                                   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 "\"Patlayıcı\" mum yönünde uzman danışman alım-satım örneği"
#property description "\"Patlayıcı\" mum k*ATR'ı aşan bir beden ölçüsü vardır"
#property description "\"Revers\" parametresi sinyal yönünü tersine çevirir"

input double lots=0.1;        // lot cinsinden hacim
input double kATR=3;          // ATR'daki sinyal mum uzunluğu
input int    ATRperiod=20;    // ATR gösterge zaman aralığı
input int    holdbars=8;      // pozisyonu açık tutmak için gereken bar sayısı
input int    slippage=10;     // izin verilebilir kayma miktarı
input bool   revers=false;    // sinyali tersine döndür?
input ulong  EXPERT_MAGIC=0;  // Uzman danışmanın MagicNumber'ı
```

```
//--- ATR gösterge yönetimini depolamak için
int atr_handle;
//--- burada son ATR değerini ve mum bedenini saklayacağız
double last_atr,last_body;
datetime lastbar_timeopen;
double trade_lot;
//+------------------------------------------------------------------+
//| Uzman danışman başlatma fonksiyonu                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- global değişkenleri başlat
   last_atr=0;
   last_body=0;
//--- doğru hacmi ayarlama
   double min_lot=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   trade_lot=lots>min_lot? lots:min_lot;
//--- ATR gösterge yönetimini oluştur
   atr_handle=iATR(_Symbol,_Period,ATRperiod);
   if(atr_handle==INVALID_HANDLE)
     {
      PrintFormat("%s: iATR oluşturulamadı, hata kodu %d",__FUNCTION__,GetLastError())
      return(INIT_FAILED);
     }
//--- başarılı uzman danışman başlatma
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Uzman danışman sonlandırma fonksiyonu                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- uzman danışman sonlandırma kodunu bildir
   Print(__FILE__,": Sonlandırma neden kodu = ",reason);
  }
//+------------------------------------------------------------------+
//| Uzman danışman fiyat(tick/tik) fonksiyonu                        |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- alım-satım sinyali
   static int signal=0; // +1 alış sinyali anlamına gelmektedir, -1 satın sinyali anla
//--- 'holdbars' bar miktarından daha önce açılmış eski pozisyonları kontrol et ve kap
   ClosePositionsByBars(holdbars,slippage,EXPERT_MAGIC);
//--- yeni bir barı kontrol et
   if(isNewBar())
     {
      //--- sinyal varlığını kontrol et
      signal=CheckSignal();
```

```
      }
//--- eğer bir netleştirme pozisyonu açıksa, sinyali atla - kapanana kadar bekle
   if(signal!=0 && PositionsTotal()>0 && (ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger
     {
      signal=0;
      return; // NewTick olay yöneticisinden çık ve yeni bir bar görünene kadar piyasa
     }
//--- bir hedging hesabı için, her bir pozisyon ayrı ayrı tutulur ve kapatılır
   if(signal!=0)
     {
      //--- alış sinyali
      if(signal>0)
        {
         PrintFormat("%s: Alış sinyali! Revers=%s",__FUNCTION__,string(revers));
         if(Buy(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
      //--- satış sinyali
      if(signal<0)
        {
         PrintFormat("%s: Satış sinyali! Revers=%s",__FUNCTION__,string(revers));
         if(Sell(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
     }
//--- OnTick fonksiyon sonlandırma
  }
//+------------------------------------------------------------------+
//| Yeni bir alım-satım sinyali kontrol et                           |
//+------------------------------------------------------------------+
int CheckSignal()
  {
//--- 0, sinyal yok anlamına gelmektedir
   int res=0;
//--- sondan bir önceki tam barın ATR değerini elde et (bar indeksi 2'dir)
   double atr_value[1];
   if(CopyBuffer(atr_handle,0,2,1,atr_value)!=-1)
     {
      last_atr=atr_value[0];
      //--- son kapanan barın verilerini bir dizi MqlRates'e ilet
      MqlRates bar[1];
      if(CopyRates(_Symbol,_Period,1,1,bar)!=-1)
        {
         //--- son kapanan barın beden ölçüsünü hesapla
         last_body=bar[0].close-bar[0].open;
         //--- eğer son barın bedeni (index 1 olan bar) bir önceki ATR değerini aşarsa
         if(MathAbs(last_body)>kATR*last_atr)
            res=last_body>0?1:-1; // yükselen mum için pozitif değer
        }
```

```
      else
         PrintFormat("%s: Son bar elde edilemedi! Hata",__FUNCTION__,GetLastError());
     }
   else
      PrintFormat("%s: ATR gösterge değeri alınamadı! Hata",__FUNCTION__,GetLastError
//--- eğer ters alım-satım modu etkinse
   res=revers?-res:res;  // eğer gerekliyse sinyali tersine çevir (1 yerine -1'i geri
//--- bir alım-satım sinyal değerini geri döndür
   return (res);
  }
//+------------------------------------------------------------------+
//|  Yeni bir bar göründüğünde 'true'yu geri döndür                  |
//+------------------------------------------------------------------+
bool isNewBar(const bool print_log=true)
  {
   static datetime bartime=0; // mevcut barın açılış zamanını sakla
//--- sıfır indeksli barın açılış zamanını elde et
   datetime currbar_time=iTime(_Symbol,_Period,0);
//--- eğer açılış zaman bilgisi değişirse, yeni bir bar gelmiş demektir
   if(bartime!=currbar_time)
     {
      bartime=currbar_time;
      lastbar_timeopen=bartime;
      //--- yeni barın açılış zamanı verilerini günlükte göster
      if(print_log && !(MQLInfoInteger(MQL_OPTIMIZATION)||MQLInfoInteger(MQL_TESTER)))
        {
         //--- yeni barın açılış zamanı bilgisine sahip bir mesaj göster
         PrintFormat("%s: yeni bar %s %s üzerinde %s de açıldı",__FUNCTION__,_Symbol,
                     StringSubstr(EnumToString(_Period),7),
                     TimeToString(TimeCurrent(),TIME_SECONDS));
         //--- son fiyat(tik) hakkındaki verileri elde et
         MqlTick last_tick;
         if(!SymbolInfoTick(Symbol(),last_tick))
            Print("SymbolInfoTick() başarısız oldu, hata = ",GetLastError());
         //--- son fiyat zamanını milisaniye olarak kadar göster
         PrintFormat("Son fiyat %s.%03d taydı",
                     TimeToString(last_tick.time,TIME_SECONDS),last_tick.time_msc%1000
        }
      //--- yeni bir bar var
      return (true);
     }
//--- yeni bir bar yok
   return (false);
  }
//+------------------------------------------------------------------+
//| Belirtilen hacimle, piyasa fiyatından alış yap                   |
//+------------------------------------------------------------------+
bool Buy(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
```

```
//--- piyasa fiyatından alış yap
   return (MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Belirtilen hacimle, piyasa fiyatından satış yap                  |
//+------------------------------------------------------------------+
bool Sell(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- piyasa fiyatından alış yap
   return (MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Bar sayısı ile hesaplanan açık tutma süresine göre pozisyonu kapat
//+------------------------------------------------------------------+
void ClosePositionsByBars(int holdtimebars,ulong deviation=10,ulong  magicnumber=0)
  {
   int total=PositionsTotal(); // açık pozisyonların sayısı
//--- tüm açık pozisyonları ara
   for(int i=total-1; i>=0; i--)
     {
      //--- pozisyon parametreleri
      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;

      //--- eğer bir pozisyonun ömrü, MagicNumber ve sembol eşleşirken, halihazırda fa
      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); // doğru mesaj formatı için metin kutusunu küçült
         PrintFormat(" #%d %s %s %.2f pozisyonunu kapat",
                     position_ticket,position_symbol,str_type,volume);
         //--- emir türünü belirle ve alım-satım isteğini gönder
         if(type==POSITION_TYPE_BUY)
            MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber,position_ticket);
         else
            MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber,position_ticket);
        }
     }
  }
//+------------------------------------------------------------------+
//| Bir alım-satım isteği hazırla ve gönder                          |
//+------------------------------------------------------------------+
bool MarketOrder(ENUM_ORDER_TYPE type,double volume,ulong slip,ulong magicnumber,ulong
```

```
  {
//--- yapıların bildirimi ve başlatımı
   MqlTradeRequest request={};
   MqlTradeResult  result={};
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(type==ORDER_TYPE_BUY)
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- istek parametreleri
   request.action   =TRADE_ACTION_DEAL;                     // İşlem türü
   request.position =pos_ticket;                            // kapanıyorsa, pozisyon k
   request.symbol   =Symbol();                              // sembol
   request.volume   =volume;                                // hacim
   request.type     =type;                                  // emir türü
   request.price    =price;                                 // alım-satım fiyatı
   request.deviation=slip;                                  // fiyattan izin verilen s
   request.magic    =magicnumber;                           // emir MagicNumber'ı
//--- bir istek gönder
   if(!OrderSend(request,result))
     {
      //--- başarısızlık verilerini görüntüle
      PrintFormat("OrderSend %s %s %.2f %.5f de hata %d",
                  request.symbol,EnumToString(type),volume,request.price,GetLastError
      return (false);
     }
//--- başarılı operasyonu bildir
   PrintFormat("retcode=%u  deal=%I64u  order=%I64u",result.retcode,result.deal,result
   return (true);
  }
```

Ayrıca bakınız

Olay yönetimi fonksiyonları, Program yütütme, Müşteri terminali olayları, OnTimer, OnBookEvent, OnChartEvent

OnCalculate

Fonksiyon, fiyat veri değişikliklerini işlemek için Calculate olayı oluştuğunda göstergelerde çağrılır. İki fonksiyon tipi vardır. Bir göstergede, sadece onlardan biri kullanılabilir.

Veri dizisine dayalı hesaplama

```
int  OnCalculate(
   const int        rates_total,       // fiyat[] dizisi boyutu
   const int        prev_calculated,   // önceki çağrıda yönetilen bar sayısı
   const int        begin,             // Anlamlı verilerin başladığı fiyat dizisi enc
   const double&    price[]            // hesaplama için değer dizisi
   );
```

Mevcut zaman aralığının zaman serilerine bağlı hesaplamalar

```
int  OnCalculate(
   const int        rates_total,       // girdi zaman serilerinin boyutu
   const int        prev_calculated,   // önceki çağrıda yönetilen bar sayısı
   const datetime&  time[],            // Açılış zamanı dizisi
   const double&    open[],            // Açılış fiyat dizisi
   const double&    high[],            // Yüksek fiyat dizisi
   const double&    low[],             // Düşük fiyat dizisi
   const double&    close[],           // Kapanış fiyat dizisi
   const long&      tick_volume[],     // Tik Hacim dizisi
   const long&      volume[],          // Gerçek hacim dizisi
   const int&       spread[]           // Spread dizisi
   );
```

Parametreler

rates_total

[in] Hesaplama için göstergeye uygun fiyat[] dizisi veya girdi serilerinin boyutu. İkinci fonksiyon tipinde; parametre değeri, çalıştığı grafikteki bar sayısına karşılık gelir.

prev_calculated

[in] Önceki çağrı sırasında, OnCalculate() fonksiyonu tarafından geri döndürülmüş değeri içerir. Bu durum, fonksiyonun önceki çalışmasından beri değişmemiş olan barların atlanması için tasarlanmıştır.

begin

[in] Anlamlı verilerin başladığı fiyat dizisi endeks değeri. Bu durum, doğru değeri olmayan kayıp veya başlangıç verilerini atlamanıza izin verir.

price[]

[in] hesaplama için değer dizisi. Fiyat zaman serilerinin biri veya bir hesaplanmış gösterge ara belleği, fiyat[] dizisi olarak geçirilebilir. Hesaplama için geçirilen veri türü _AppliedTo önceden tanımlanmış değişkeni kullanılarak tanımlanabilir.

time[]

[in] Bar açılış zaman değerlerini içeren dizi.

open[]

[in] Açılış(Open) fiyat değerlerini içeren dizi.

high[]

[in] Yüksek(High) fiyat değerlerini içeren dizi.

low[]

[in] Düşük(Low) fiyat değerlerini içeren dizi.

close[]

[in] Kapanış(Close) fiyat değerlerini içeren dizi.

tick_volume[]

[in] Tik hacim değerlerini içeren dizi.

volume[]

[in] Alım-satım hacim(Real volume) değerlerini içeren dizi.

spread[]

[in] Barların spread değerlerini içeren dizi.

Geri dönüş değeri

Bir sonraki fonksiyon çağrısı sırasında *prev_calculated* parametresi olarak geçirilecek int tip değeri.

Not

Eğer OnCalculate() fonksiyonu sıfıra eşitse, müşteri terminalinin DataWindow bölümünde hiçbir gösterge değeri gösterilmez.

Eğer fiyat verisi OnCalculate() fonksiyonunun son çağrısından beri değiştirilmişse (daha derin bir geçmiş yüklendi veya geçmişteki fiyat boşluğu doldu), *prev_calculated* girdi parametresinin değeri terminalin kendisi tarafından sıfıra ayarlanır.

time[], *open[]*, *high[]*, *low[]*, *close[]*, *tick_volume[]*, *volume[]* ve *spread[]* dizileri arasında indeksleme yönünü belirlemek için, ArrayGetAsSeries() fonksiyonunu çağırın. Varsayılanlara bağlı olmamak için, ArraySetAsSeries() fonksiyonunu çalışılacak olan diziler için çağırın.

İlk fonksiyon tipinin kullanımında; gösterge çalıştırılırken parametre sekmesindeki -gerekli zaman serileri yada gösterge- ayarı, bir kullanıcı tarafından fiyat[] dizisi olarak seçilir. Bunu yapmak için, "Apply to" alanının açılır listesinde gerekli elemanları belirtin.

Diğer MQL5 programlarından özel gösterge değerlerini almak için, iCustom() fonksiyonu kullanılır. Sonraki operasyonlar için gösterge yönetimini geri döndürür. Ayrıca, gerekli *fiyat []* dizisini veya farklı bir göstergenin yönetimini belirtmekte mümkündür. Bu parametre, bir özel göstergenin girdi değişkenleri listesinde en son geçmelidir.

OnCalculate() fonksiyonu tarafından ve *prev_calculated* ikinci girdi parametresi tarafından geri döndürülen değerler arasındaki bağlantıyı kullanmak gereklidir. Fonksiyonu çağırırken, *prev_calculated* parametresi bir önceki çağrı sırasında OnCalculate() fonksiyonu tarafından geri döndürülen değeri içerir. Bu durum, fonksiyonun bir önceki çalışmasından beri değişmemiş olan barlar için tekrarlayan hesaplamaları önlemek amacıyla, özel bir göstergenin hesaplanması adına kaynak koruyucu algoritmaların uygulanmasını mümkün kılar.

Örnek gösterge

```
//+------------------------------------------------------------------+
//|                                            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 "Örnek Momentum göstergesi hesaplaması"

//---- gösterge ayarları
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
#property indicator_type1   DRAW_LINE
#property indicator_color1  Blue
//---- girdiler
input int MomentumPeriod=14; // Hesaplama zaman aralığı
//---- gösterge arabelleği
double    MomentumBuffer[];
//--- Hesaplama zaman aralığını saklamak için global değişkenler
int       IntPeriod;
//+------------------------------------------------------------------+
//| Özel gösterge başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- girdi parametrelerini kontrol et
   if(MomentumPeriod<0)
     {
      IntPeriod=14;
      Print("Zaman aralığı parametresi yanlış bir değere sahiptir. Aşağıdaki değer hes
     }
   else
      IntPeriod=MomentumPeriod;
//---- arabellekler
   SetIndexBuffer(0,MomentumBuffer,INDICATOR_DATA);
//---- DataWindow ve alt pencerede görüntülenecek gösterge ismi
   IndicatorSetString(INDICATOR_SHORTNAME,"Momentum"+"("+string(IntPeriod)+")");
//--- çizilmeye oradan başlanacak olan barın indeksini ayarla
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,IntPeriod-1);
//--- çizilmeyen boş bir değer olarak 0.0'ı ayarla
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
//--- görüntülenecek gösterge hassasiyeti
   IndicatorSetInteger(INDICATOR_DIGITS,2);
  }
//+------------------------------------------------------------------+
```

```
//|  Momentum gösterge hesaplaması                                   |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,     // fiyat[] dizisi boyutu
                const int prev_calculated, // önceki yönetilen bar sayısı
                const int begin,           // anlamlı verilerin başladığı yer
                const double &price[])     // yönetim için veri dizisi
  {
//--- hesaplamalar için başlangıç pozisyonu
   int StartCalcPosition=(IntPeriod-1)+begin;
//---- eğer hesaplama verisi yetersizse
   if(rates_total<StartCalcPosition)
      return(0);  // sıfır değeri ile çık - gösterge hesaplanmaz
//--- doğru çizim başlar
   if(begin>0)
      PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,StartCalcPosition+(IntPeriod-1));
//--- hesaplamayı başlat, başlangıç pozisyonunu tanımla
   int pos=prev_calculated-1;
   if(pos<StartCalcPosition)
      pos=begin+IntPeriod;
//--- ana hesaplama döngüsü
   for(int i=pos;i<rates_total && !IsStopped();i++)
      MomentumBuffer[i]=price[i]*100/price[i-IntPeriod];
//--- OnCalculate gerçekleşimi tamamlandı. Sonraki çağrı için yeni prev_calculated değ
   return(rates_total);
  }
```

Ayrıca bakınız

ArrayGetAsSeries, ArraySetAsSeries, iCustom, Olay yönetimi fonksiyonları, Program yürütme, Müşteri terminali olayları, Zaman serilerine ve göstergelere eriş

OnTimer

Fonksiyon, Timer olayının terminal tarafından sabit zaman aralıklarında oluşturulmasıyla uzman danışmanlarda çağrılır.

```
void  OnTimer(void);
```

Geri dönüş değeri

Geri dönüş değeri yok

Not

Timer olayı, EventSetTimer() fonksiyonu ile zamanlayıcıyı etkin hale getiren bir uzman danışman için, müşteri terminali tarafından periyodik olarak oluşturulur. Genellikle, bu fonksiyon OnInit() fonksiyonu içerisinde çağrılır. Uzman danışman çalışmayı durdurduğunda, zamanlayıcı genellikle OnDeinit() fonksiyonu içerisinde çağrılan EventKillTimer() fonksiyonu ile kaldırılır.

Her bir Uzman Danışman ve her bir gösterge kendi zamanlayıcısı ile çalışır ve olayları sadece ondan alır. Mql5 uygulamasının kapatılması sırasında; zamanlayıcı, önceden oluşturulup ancak devamında EventKillTimer() fonksiyonu ile devre dışı bırakılmaması durumunda, zorla yok edilir.

Eğer zamanlayıcı olaylarını saniyede bir kereden daha sık almanız gerekiyorsa, yüksek çözünürlüklü bir zamanlayıcı oluşturmak için EventSetMillisecondTimer() fonksiyonunu kullanın.

Genelde, zamanlayıcı zaman aralığı azaltıldığında; zamanlayıcı olaylarının yöneticisi daha sık çağrıldığından, test süresi artar. Gerçek zamanlı modda çalışırken, zamanlayıcı olayları, donanım sınırlamaları nedeniyle 10-16 milisaniye içinde 1 kereden daha fazla üretilemez.

Her bir program için sadece bir zamanlayıcı çalıştırılabilir. Her bir MQL5 uygulaması ve grafiği, yeni gelen tüm olayların yerleştiği kendisine ait olaylar sırasına(kuyruğuna) sahiptir. Eğer sıra zaten Timer olayını içeriyorsa veya bu olay yönetilme aşamasındaysa, o zaman yeni Timer olayı MQL5 uygulama sırasına eklenmez.

OnTimer() yöneticisiyle **Örnek Uzman Danışman**

```
//+------------------------------------------------------------------+
//|                                                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 "Alım-satım sunucusu zamanını hesaplamak için zamanlayıcı kullar
#property description "Uzman danışmanı, hafta sonundan önce bir işlem haftasının sonur
//+------------------------------------------------------------------+
//| Uzman danışman başlatma fonksiyonu                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- 1 saniye zaman aralıklı bir zamanlayıcı oluştur
   EventSetTimer(1);
```

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Uzman danışman sonlandırma fonksiyonu                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- işi tamamladıktan sonra zamanlayıcıyı kaldır
   EventKillTimer();

  }
//+------------------------------------------------------------------+
//| Uzman danışman tik fonksiyonu                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Zamanlayıcı fonksiyonu                                           |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- OnTimer() ilk çağrısının zamanı
   static datetime start_time=TimeCurrent();
//--- İlk OnTimer() çağrısı sırasında alım-satım sunucusu zamanı
   static datetime start_tradeserver_time=0;
//--- hesaplanmış alım-satım sunucusu zamanı
   static datetime calculated_server_time=0;
//--- yerel bilgisayar zamanı
   datetime local_time=TimeLocal();
//--- mevcut tahmini alım-satım sunucusu zamanı
   datetime trade_server_time=TimeTradeServer();
//--- eğer sunucu zamanı bir nedenden dolayı bilinmiyorsa, vaktinden önce çıkın
   if(trade_server_time==0)
      return;
//--- eğer başlangıç alım-satım sunucusu değeri henüz belirlenmemişse
   if(start_tradeserver_time==0)
     {
      start_tradeserver_time=trade_server_time;
      //--- alım-satım sunucusunun hesaplanmış değerini ayarla
      Print(trade_server_time);
      calculated_server_time=trade_server_time;
     }
   else
     {
```

```
      //--- OnTimer() ilk çağrısının süresini artır
      if(start_tradeserver_time!=0)
         calculated_server_time=calculated_server_time+1;;
     }
//---
   string com=StringFormat("                  Başlangıç zamanı: %s\r\n",TimeToString(s
   com=com+StringFormat("                  Yerel zaman: %s\r\n",TimeToString(local_tir
   com=com+StringFormat("TimeTradeServer zamanı: %s\r\n",TimeToString(trade_server_tir
   com=com+StringFormat(" EstimatedServer zamanı: %s\r\n",TimeToString(calculated_serv
//--- grafikteki tüm sayaçların değerlerini görüntüle
   Comment(com);
  }
```

Ayrıca bakınız

EventSetTimer, EventSetMillisecondTimer, EventKillTimer, GetTickCount, GetMicrosecondCount, Müşteri terminali olayları

OnTrade

Fonksiyon, Trade olayı meydana geldiğinde uzman danışmanlarda çağrılır. Fonksiyon; emir, pozisyon ve işlem listesindeki değişiklikleri işlemek içindir.

```
void  OnTrade(void);
```

Geri dönüş değeri

Geri dönüş değeri yok

Not

OnTrade() sadece uzman danışmanlar için çağrılır. Göstergeler ve komut dosyalarında, aynı isme ve türe sahip bir fonksiyon ekleseniz bile, kullanılamaz.

Herhangi bir alım-satım eylemi (bekleyen emir yerleştirme, bir pozisyon açma/kapama, zararı durdurları yerleştirme, bekleyen emirleri aktifleştirme, vb.) için , emirlerin ve işlemlerin geçmişi ve/veya pozisyonların ve mevcut emirlerin listesi uygun olarak değiştirilir.

Bir emri yönetirken, bir alım satım sunucusu terminale, gelmekte olan Trade olayı hakkında mesaj gönderir. Geçmişten emirler ve işlemlerle ilgili verileri elde etmek için, öncelikle HistorySelect()'i kullanarak bir işlem geçmişi isteği gerçekleştirmek gerekir.

Alım-satım olayları sunucu tarafından şu durumlarda oluşturulur:

- aktif emirleri değiştirme,
- pozisyonları değiştirmek,
- işlemleri değiştirme
- işlem geçmişini değiştirme.

Her bir Trade olayı bir veya birkaç alım-satım işlemi isteğinin sonucunda görülebilir. İşlem istekleri sunucuya OrderSend() veya OrderSendAsync() fonksiyonları kullanılarak gönderilir. Her bir istek, birkaç alım-satım olayına neden olabilir. Olayların işlenmesi birkaç aşamada gerçekleştirilebildiğinden ve her operasyon emirlerin, pozisyonların ve işlem geçmişinin durumunu değiştirebileceğinden, "Bir istek - bir Trade olayı" ifadesine güvenemezsiniz.

OnTrade() yöneticisi, uygun OnTradeTransaction() çağrılarından sonra çağrılır. Genelde, OnTrade () ve OnTradeTransaction () çağrılarının sayısında tam bir korelasyon yoktur. Bir OnTrade() çağrısı, bir veya birkaç OnTradeTransaction çağrısına karşılık gelir.

OnTrade() yöneticisiyle **Örnek Uzman Danışman**

```
//+------------------------------------------------------------------+
//|                                                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;             // günlerde alım-satım geçmişinin derinliği
```

```
//--- alım-satım geçmişinin sınırlarını küresel kapsamda belirle
datetime     start;             // önbellekteki alım-satım geçmişi için başlangıç zama
datetime     end;               // önbellekteki alım-satım geçmişi için bitiş zamanı
//--- global sayaçlar
int          orders;            // aktif emirlerin sayısı
int          positions;         // açık pozisyonların sayısı
int          deals;             // alım-satım geçmişi önbelleğinde işlemlerin sayısı
int          history_orders;    // alım-satım geçmişi önbelleğinde emirlerin sayısı
bool         started=false;     // sayaç ilgi bayrağı

//+------------------------------------------------------------------+
//| Uzman danışman başlatma fonksiyonu                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   end=TimeCurrent();
   start=end-days*PeriodSeconds(PERIOD_D1);
   PrintFormat("Yüklenebilir alım-satım geçmişin sınırları: başlangıç - %s, bitiş - %s
               TimeToString(start),TimeToString(end));
   InitCounters();
//---
   return(0);
  }
//+------------------------------------------------------------------+
//|  Pozisyon, emir ve işlem sayaçlarının başlatılması               |
//+------------------------------------------------------------------+
void InitCounters()
  {
   ResetLastError();
//--- geçmişi yükle
   bool selected=HistorySelect(start,end);
   if(!selected)
     {
      PrintFormat("%s. Önbellek geçmişini %s ten %s konumuna yükleme başarısız oldu. E
                  __FUNCTION__,TimeToString(start),TimeToString(end),GetLastError());
      return;
     }
//--- Mevcut değeri elde edin
   orders=OrdersTotal();
   positions=PositionsTotal();
   deals=HistoryDealsTotal();
   history_orders=HistoryOrdersTotal();
   started=true;
   Print("emirlerin, pozisyonların ve işlemlerin sayaçları başarılı bir şekilde başlat
  }
//+------------------------------------------------------------------+
//| Uzman danışman tik fonksiyonu                                    |
//+------------------------------------------------------------------+
```

```
void OnTick()
  {
   if(started) SimpleTradeProcessor();
   else InitCounters();
  }
//+------------------------------------------------------------------+
//| bir Trade olayı gerçekleştiğinde çağrılır                        |
//+------------------------------------------------------------------+
void OnTrade()
  {
   if(started) SimpleTradeProcessor();
   else InitCounters();
  }
//+------------------------------------------------------------------+
//| alım-satım işlemlerindeki ve geçmişteki değişiklikleri işlemenin örneği
//+------------------------------------------------------------------+
void SimpleTradeProcessor()
  {
   end=TimeCurrent();
   ResetLastError();
//--- işlem geçmişini belirtilen aralıktan program önbelleğine yükle
   bool selected=HistorySelect(start,end);
   if(!selected)
     {
      PrintFormat("%s. Önbellek geçmişini %s ten %s konumuna yükleme başarısız oldu. E
                  __FUNCTION__,TimeToString(start),TimeToString(end),GetLastError());
      return;
     }
//--- mevcut değerleri elde et
   int curr_orders=OrdersTotal();
   int curr_positions=PositionsTotal();
   int curr_deals=HistoryDealsTotal();
   int curr_history_orders=HistoryOrdersTotal();
//--- aktif emirlerin sayısının değişip değişmediğini kontrol edin
   if(curr_orders!=orders)
     {
      //--- aktif emirlerin sayısı değişti
      PrintFormat("emirlerin sayısı değişti. Önceki değer %d, şu anki is %d dir",
                  orders,curr_orders);
      //--- değeri güncelle
      orders=curr_orders;
     }
//--- açık pozisyonların sayısındaki değişiklikler
   if(curr_positions!=positions)
     {
      //--- açık pozisyonların sayısı değişti
      PrintFormat("Pozisyonların sayısı değişti. Önceki değer %d, şu anki is %d dir",
                  positions,curr_positions);
      //--- değeri güncelle
```

```
      positions=curr_positions;
     }
//--- işlem geçmişi önbelleğindeki emirlerin sayısındaki değişiklikler
   if(curr_deals!=deals)
     {
      //--- işlem geçmişi önbelleğindeki emirlerin sayısı değişti
      PrintFormat("İşlemlerin sayısı değişti. Önceki değer %d, şu anki is %d dir",
                  deals,curr_deals);
      //--- değeri güncelle
      deals=curr_deals;
     }
//--- işlem geçmişi önbelleğindeki geçmiş emirlerin sayısındaki değişiklikler
   if(curr_history_orders!=history_orders)
     {
      //--- işlem geçmişi önbelleğindeki geçmiş emirlerin sayısı değişti
      PrintFormat("Geçmişteki emirlerin sayısı değişti. Önceki değer %d, şu anki is %d
                  history_orders,curr_history_orders);
     //--- değeri güncelle
     history_orders=curr_history_orders;
     }
//--- önbellekte istenecek işlem geçmişinin sınırlarının değiştirilmesinin gerekli olu
   CheckStartDateInTradeHistory();
  }
//+------------------------------------------------------------------+
//|  işlem geçmişini istemek için başlangıç zamanını değiştirme      |
//+------------------------------------------------------------------+
void CheckStartDateInTradeHistory()
  {
//--- başlangıç aralığı, şu an çalışmaya başlayacak olsaydık
   datetime curr_start=TimeCurrent()-days*PeriodSeconds(PERIOD_D1);
//--- işlem geçmişinin başlangıç sınırının gitmediğinden emin ol
//--- hedeflenen tarihten 1 günden fazla
   if(curr_start-start>PeriodSeconds(PERIOD_D1))
     {
      //--- önbelleğe yüklenecek geçmişin başlangıç tarihini doğrula
      start=curr_start;
      PrintFormat("Yüklenecek işlem geçmişinin yeni başlangıç sınırı: başlangıç => %s"
                  TimeToString(start));
      //--- şimdi güncellenmiş aralık için işlem geçmişini yeniden yükle
      HistorySelect(start,end);
      //--- ilerki karşılaştırma için geçmişteki işlem ve emir sayaçlarını doğrula
      history_orders=HistoryOrdersTotal();
      deals=HistoryDealsTotal();
     }
  }
//+------------------------------------------------------------------+
/* Örnek çıktı:
  Yüklenecek geçmişin limiti: başlangıç - 2018.07.16 18:11, bitiş - 2018.07.23 18:11
  Emirlerin, pozisyonların ve işlemlerin sayaçları başarılı bir şekilde başlatıldı
```

```
   Emirlerin sayısı değişti. Önceki değer 0, şu anki değer 1
   Emirlerin sayısı değişti. Previous value 1, current value 0
   Pozisyonların sayısı değişti. Önceki değer 0, şu anki değer 1
   İşlemlerin sayısı değişti. Önceki değer 0, şu anki değer 1
   Geçmişteki emirlerin sayısı değişti. Önceki değer 0, şu anki değer 1
*/
```

Ayrıca bakınız

OrderSend, OrderSendAsync, OnTradeTransaction, Müşteri terminali olayları

OnTradeTransaction

Fonksiyon, TradeTransaction olayı meydana geldiğinde uzman danışmanlarda çağrılır. Fonksiyon, alım-satım işlem isteği gerçekleşimi sonuçlarını yönetmek içindir.

```
void  OnTradeTransaction()
   const MqlTradeTransaction&    trans,     // alım-satım işlem yapısı
   const MqlTradeRequest&        request,   // istek yapısı
   const MqlTradeResult&         result     // cevap yapısı
   );
```

Parametreler

trans

[in] Bir alım-satım hesabında yapılan bir işlemi tanımlayan MqlTradeTransaction tipi değişken.

request

[in] Bir alım-satım isteğini (bir işlemi meydana getiren) tanımlayan MqlTradeRequest tipi değişken. Sadece TRADE_TRANSACTION_REQUEST tipi işlem için değerler içerir.

result

[in] Bir alım-satım isteğinin (bir işlemi meydana getiren) gerçekleşim sonucunu içeren MqlTradeResult tipi değişken. Sadece TRADE_TRANSACTION_REQUEST tipi işlem için değerler içerir.

Geri dönüş değeri

Geri dönüş değeri yok

Not

OnTradeTransaction(), bir işlem sunucusu tarafından terminale gönderilen TradeTransaction olayını yönetmek için aşağıdaki durumlarda çağrılır:

- OrderSend()/OrderSendAsync() fonksiyonlarını ve devamında gerçekleşimini kullanarak bir MQL5 programından bir işlem isteği gönderimi;
- grafik kullanıcı arayüzü (GUI) ve devamında gerçekleşim ile manuel olarak bir işlem isteği gönderme;
- sunucu üzerindeki bekleyen emirler ve zararı durdur emirlerinin aktifleştirilmesi;
- işlem sunucusu tarafında operasyonlar gerçekleştirme.

İşlem türüne ilişkin veriler, *trans* değişkeninin *type* alanında bulunur. Alım-satım işlem türleri, ENUM_TRADE_TRANSACTION_TYPE listesinde tanımlanmaktadır:

- TRADE_TRANSACTION_ORDER_ADD - yeni bir aktif emir ekleme
- TRADE_TRANSACTION_ORDER_UPDATE - varolan bir emri değiştirme
- TRADE_TRANSACTION_ORDER_DELETE - aktif olan emirlerden birini silme
- TRADE_TRANSACTION_DEAL_ADD - geçmişe bir işlem ekleme
- TRADE_TRANSACTION_DEAL_UPDATE - geçmişteki bir işlemi değiştirme
- TRADE_TRANSACTION_DEAL_DELETE - geçmişten bir işlemi silme
- TRADE_TRANSACTION_HISTORY_ADD - gerçekleşim veya iptal etme sonucunda geçmişe bir emri ekleme
- TRADE_TRANSACTION_HISTORY_UPDATE - emir geçmişindeki bir emri değiştirme

- TRADE_TRANSACTION_HISTORY_DELETE - emir geçmişinden bir emri silme
- TRADE_TRANSACTION_POSITION - bir işlem gerçekleşimi ile ilgili olmayan pozisyon değişikliği
- TRADE_TRANSACTION_REQUEST - bir alım-satım isteğinin sunucu tarafından işlendiğini ve de bu işlemin sonucunun elde edildiğine dair bildirim.

TRADE_TRANSACTION_REQUEST tipi işlemleri yönetirken, ek bilgi edinmek için OnTradeTransaction() fonksiyonunun ikinci ve üçüncü parametrelerini - *request* ve *result* - analiz etmek gereklidir.

Bir alım-satım hesabında bir işlem isteği göndermek, alım-satım işlemler zincirine yol açar: 1) istek, işlenme için kabul edilir, 2) hesap için uygun bir satınalma emri oluşturulur, 3) emir devamında gerçekleştirilir, 4) gerçekleştirilen emir, aktif olan emirlerin listesinden silinir, 5) emir geçmişine eklenir, 6) bir sonraki işlem geçmişe eklenir ve 7) yeni bir pozisyon oluşturulur. Tüm bu aşamalar alım-satım işlemleri dir. Terminale ulaşan bunun gibi her bir işlem TradeTransaction olayıdır. Terminalde bu işlemlerin geliş önceliği garanti edilemez. Bu nedenle, işlem algoritmanızı geliştirirken bir grup işlemin diğerinden sonra gelmesini beklememelisiniz.

İşlemler uzman danışmanın OnTradeTransaction() yöneticisi tarafından işlenirken, terminal gelen alım-satım işlemlerini yönetmeye devam eder. Bundan dolayı, işlem hesabının durumu OnTradeTransaction() operasyonu sırasında değişebilir. Örneğin, bir MQL5 programı eklenen yeni bir emri yönetirken, emir gerçekleştirilebilir, açık emirler listesinden silinebilir ve geçmişe taşınabilir. Program tüm bu olaylardan haberdar edilir.

İşlemler kuyruk uzunluğu 1024 elemandan oluşur. Eğer OnTradeTransaction() başka bir işlemi çok uzun süre yönetiyorsa, kuyruktaki yeni işlemler önceki işlemlerin yerine geçebilir.

OnTrade() yöneticisi, uygun OnTradeTransaction() çağrılarından sonra çağrılır. Genelde, OnTrade () ve OnTradeTransaction () çağrılarının sayısında tam bir korelasyon yoktur. Bir OnTrade() çağrısı, bir veya birkaç OnTradeTransaction çağrısına karşılık gelir.

Her bir Trade olayı bir veya birkaç işlem isteğinin sonucunda görülebilir. İşlem istekleri, OrderSend() veya OrderSendAsync() kullanılarak sunucuya gönderilir. Her bir istek, birkaç alım-satım olayına neden olabilir. Olayların işlenmesi birkaç aşamada gerçekleştirilebildiğinden ve her operasyon emirlerin, pozisyonların ve işlem geçmişinin durumunu değiştirebileceğinden, "Bir istek - bir Trade olayı" ifadesine güvenemezsiniz.

OnTradeTransaction() yöneticisiyle **Örnek Uzman Danışman**

```
//+------------------------------------------------------------------+
//|                                    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 "TradeTransaction olayının örnek dinleyicisi"
//+------------------------------------------------------------------+
//| Uzman danışman başlatma fonksiyonu                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
```

```
   PrintFormat("SON PING=%.f ms",
               TerminalInfoInteger(TERMINAL_PING_LAST)/1000.);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Uzman danışman tik fonksiyonu                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| TradeTransaction fonksiyonu                                      |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//---
   static int counter=0;   // OnTradeTransaction() çağrılarının sayacı
   static uint lasttime=0; // OnTradeTransaction() son çağrısının zamanı
//---
   uint time=GetTickCount();
//--- eğer son işlem 1 saniyeden daha önce meydana geldiyse,
   if(time-lasttime>1000)
     {
      counter=0; // o zaman bu, yeni bir işlem operasyonudur, sayaç sıfırlanabilir
      if(IS_DEBUG_MODE)
         Print(" Yeni işlem operasyonu");
     }
   lasttime=time;
   counter++;
   Print(counter,". ",__FUNCTION__);
//--- işlem isteği gerçekleşiminin sonucu
   ulong            lastOrderID   =trans.order;
   ENUM_ORDER_TYPE  lastOrderType =trans.order_type;
   ENUM_ORDER_STATE lastOrderState=trans.order_state;
//--- işlemin gerçekleştiği sembolün adı
   string trans_symbol=trans.symbol;
//--- işlemin türü
   ENUM_TRADE_TRANSACTION_TYPE  trans_type=trans.type;
   switch(trans.type)
     {
      case  TRADE_TRANSACTION_POSITION:    // pozisyonda değişiklik
        {
         ulong pos_ID=trans.position;
         PrintFormat("MqlTradeTransaction: Pozisyon  #%d %s değişti: SL=%.5f TP=%.5f",
```

```
                  pos_ID,trans_symbol,trans.price_sl,trans.price_tp);
     }
   break;
   case TRADE_TRANSACTION_REQUEST:     // bir işlem isteği gönderme
      PrintFormat("MqlTradeTransaction: TRADE_TRANSACTION_REQUEST");
      break;
   case TRADE_TRANSACTION_DEAL_ADD:    // bir işlem ekleme
     {
      ulong          lastDealID   =trans.deal;
      ENUM_DEAL_TYPE lastDealType =trans.deal_type;
      double         lastDealVolume=trans.volume;
      //--- Dahili bir sistemde işlem kimliği - bir döviz tarafından atanan etiket
      string Exchange_ticket="";
      if(HistoryDealSelect(lastDealID))
         Exchange_ticket=HistoryDealGetString(lastDealID,DEAL_EXTERNAL_ID);
      if(Exchange_ticket!="")
         Exchange_ticket=StringFormat("(Döviz işlemi=%s)",Exchange_ticket);

      PrintFormat("MqlTradeTransaction: %s işlem #%d %s %s %.2f lot   %s",EnumToStr
                  lastDealID,EnumToString(lastDealType),trans_symbol,lastDealVolume
     }
   break;
   case TRADE_TRANSACTION_HISTORY_ADD: // geçmişe bir emir ekleme
     {
      //--- Dahili bir sistemde işlem kimliği - bir Döviz tarafından atanan etiket
      string Exchange_ticket="";
      if(lastOrderState==ORDER_STATE_FILLED)
        {
         if(HistoryOrderSelect(lastOrderID))
            Exchange_ticket=HistoryOrderGetString(lastOrderID,ORDER_EXTERNAL_ID);
         if(Exchange_ticket!="")
            Exchange_ticket=StringFormat("(Döviz etiketi=%s)",Exchange_ticket);
        }
      PrintFormat("MqlTradeTransaction: %s emir #%d %s %s %s   %s",EnumToString(tra
                  lastOrderID,EnumToString(lastOrderType),trans_symbol,EnumToStrin
     }
   break;
   default: // diğer işlemler
     {
      //--- Dahili bir sistemde işlem kimliği - Moskova Döviz tarafından atanan eti
      string Exchange_ticket="";
      if(lastOrderState==ORDER_STATE_PLACED)
        {
         if(OrderSelect(lastOrderID))
            Exchange_ticket=OrderGetString(ORDER_EXTERNAL_ID);
         if(Exchange_ticket!="")
            Exchange_ticket=StringFormat("Döviz etiketi=%s",Exchange_ticket);
        }
      PrintFormat("MqlTradeTransaction: %s emir #%d %s %s   %s",EnumToString(trans_
```

```
                     lastOrderID,EnumToString(lastOrderType),EnumToString(lastOrderSta
        }
       break;
      }
//--- emir etiketi
   ulong orderID_result=result.order;
   string retcode_result=GetRetcodeID(result.retcode);
   if(orderID_result!=0)
      PrintFormat("MqlTradeResult: emir #%d retcode=%s ",orderID_result,retcode_result
//---
  }
//+------------------------------------------------------------------+
//| sayısal yanıt kodlarını, dizi yapısında anımsanır kodlara dönüştür
//+------------------------------------------------------------------+
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;
     }
//---
  }
```

Ayrıca bakınız

OrderSend, OrderSendAsync, OnTradeTransaction, İşlem isteği yapısı, Alım-satım işlem yapısı, Alım-satım işlem türleri, İşlem operasyon türü, Müşteri terminal olayları

OnBookEvent

Fonksiyon, BookEvent olayı meydana geldiğinde göstergelerde ve uzman danışmanlarda çağrılır. Piyasa Derinliği değişikliklerini yönetmek içindir.

```
void  OnBookEvent(
   const string&  symbol          // sembol
   );
```

Parametreler

symbol

[in] BookEvent in geldiği sembolün adı

Geri dönüş değeri

Geri dönüş değeri yok

Not

Herhangi bir sembol için BookEvent olaylarını almak için, MarketBookAdd() fonksiyonunu kullanarak ilgili sembollere abone olun. İlgili sembol için BookEvent olaylarını almak amacıyla açılan aboneliği iptal etmek için, MarketBookRelease() fonksiyonunu çağırın.

BookEvent, tüm grafikte yayınlanır. Bu durum; bir grafikteki bir uygulamanın MarketBookAdd fonksiyonunu kullanarak BookEvent e abone olması durumunda, aynı grafikte başlatılan ve OnBookEvent() yöneticisine sahip olan diğer tüm göstergeler ve uzman danışmanların da bu olayı alması anlamına gelmektedir Bu nedenle, OnBookEvent() yöneticisine aktarılan bir sembol adını, *symbol* parametresi olarak analiz etmek gerekir.

Aynı grafikte çalışan tüm uygulamalar için, sembollere göre sıralanmış ayrı BookEvent sayaçları sağlanır. Bu, her bir grafiğin farklı sembollerde çoklu aboneliklere sahip olabileceği ve her sembol için bir sayacın sağlandığı anlamına gelir. BookEvent e abone olmak ve abonelikten çıkmak, belirtilen semboller için abonelik sayacını yalnızca bir grafikte değiştirir. Başka bir deyişle, aynı sembol için BookEvent e bitişik iki grafik olabilir, ancak farklı abonelik sayaç değerlerine sahip olabilirler.

Başlangıç abonelik sayaç değeri sıfırdır. Her bir MarketBookAdd() çağrısında, grafikte belirtilen bir sembol için abonelik sayacı bir arttırılır (MarketBookAdd()'deki grafik ve sembol eşleşmek zorunda değildir). MarketBookRelease() i çağırırken, grafikteki belirli bir sembol için abonelik sayacı bir azalır. Herhangi bir sembol için BookEvent olayları, sayaç sıfıra eşit olana kadar grafikte yayınlanır. Bu nedenle, çalışmasının sonunda MarketBookAdd () çağrılarını içeren her bir MQL5 programının, MarketBookRelease () i kullanarak her sembol için olay alımı aboneliğinden doğru bir şekilde ayrılması önemlidir. Bunu başarmak için, MarketBookAdd() ve MarketBookRelease() çağrılarının sayısı, MQL5 program ömrünün tamamı boyunca her çağrı için bile olmalıdır. Program içindeki bayrakların veya özel abonelik sayaçlarının kullanılması, BookEvent olaylarıyla güvenle çalışmanızı sağlar ve bu olayı aynı grafikteki üçüncü parti programlarında da almak için aboneliklerin devre dışı bırakılmasını engeller.

BookEvent olayları hiçbir zaman atlanmaz, bir önceki BookEvent yönetimi hala sonlanmamış olasa bile her zaman kuyruğa eklenir.

Örnek

```
//+------------------------------------------------------------------+
//|                                          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 "OnBookEvent() i kullanarak piyasa derinliği yenilenme oranının
#property description "Kod şu makaleden alınmıştır: https://www.mql5.com/en/articles/2
//--- girdi parametreleri
input ulong ExtCollectTime   =30;  // saniye cinsinden test süresi
input ulong ExtSkipFirstTicks=10;  // başlangıçta atlanan tik sayısı
//--- BookEvent olaylarına aboneliğin bayrağı
bool book_subscribed=false;
//--- piyasa derinliğinden istekleri kabul etmek için dizi
MqlBookInfo  book[];
//+------------------------------------------------------------------+
//| Uzman danışman başlatama fonksiyonu                              |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- başlangıcı göster
   Comment(StringFormat("İlk %I64u ticklerinin gelmesini bekleme",ExtSkipFirstTicks));
   PrintFormat("İlk %I64u ticklerinin gelmesini bekleme",ExtSkipFirstTicks);
//--- piyasa derinliği yayınlanımını etkinleştir
   if(MarketBookAdd(_Symbol))
     {
      book_subscribed=true;
      PrintFormat("%s: MarketBookAdd(%s) fonksiyonu true geri döndürdü",__FUNCTION__,
     }
   else
      PrintFormat("%s: MarketBookAdd(%s) fonksiyonu false geri döndürdü! GetLastError
//--- başarılı başlatma
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Uzman danışmanı sonlandır                                        |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- sonlandırma neden kodunu görüntüle
   Print(__FUNCTION__,": Sonlandırma neden kodu = ",reason);
//--- piyasa derinliği olayları alınımı aboneliğini iptal et
   if(book_subscribed)
     {
      if(!MarketBookRelease(_Symbol))
         PrintFormat("%s: MarketBookRelease(%s) false geri döndürdü! GetLastError()=%
```

```
      else
         book_subscribed=false;
     }
//---
  }
//+------------------------------------------------------------------+
//| BookEvent fonksiyonu                                             |
//+------------------------------------------------------------------+
void OnBookEvent(const string &symbol)
  {
   static ulong starttime=0;              // test başlangıç zamanı
   static ulong tickcounter=0;            // piyasa derinliği güncelleme sayacı
//--- sadece eğer onlara abone olunmuşsa piyasa derinliği olayları ile çalış
   if(!book_subscribed)
      return;
//--- sadece belirli bir sembol için güncellemeleri say
   if(symbol!=_Symbol)
      return;
//--- kuyruğu temizlemek ve hazırlamak için ilk tikleri atlayın
   tickcounter++;
   if(tickcounter<ExtSkipFirstTicks)
      return;
//--- başlangıç zamanını hatırla
   if(tickcounter==ExtSkipFirstTicks)
      starttime=GetMicrosecondCount();
//--- piyasa derinliği verileri için istek oluştur
   MarketBookGet(symbol,book);
//--- ne zaman durulacak?
   ulong endtime=GetMicrosecondCount()-starttime;
   ulong ticks  =1+tickcounter-ExtSkipFirstTicks;
// testin başlangıcından bu yana mikrosaniye cinsinden ne kadar süre geçti?
   if(endtime>ExtCollectTime*1000*1000)
     {
      PrintFormat("%.1f saniye için %I64u tikler: %.1f tikler/saniye ",ticks,endtime/1
      ExpertRemove();
      return;
     }
//--- yorum alanında sayaçları görüntüle
   if(endtime>0)
      Comment(StringFormat("%.1f saniye için %I64u tikler: %.1f tikler/saniye ",ticks,
  }
```

Ayrıca bakınız

MarketBookAdd, MarketBookRelease, MarketBookGet, OnTrade, OnTradeTransaction, OnTick, Olay yönetimi fonksiyonları, Program yürütme, Müşteri terminal olayları

OnChartEvent

Fonskiyon, ChartEvent olayı meydana geldiğinde göstergeler ve uzman danışmanlarda çağrılır. Fonksiyon; grafikte, bir kullanıcı veya bir MQL5 programı tarafından yapılan değişiklikleri yönetmek içindir.

```
void  OnChartEvent()
   const int       id,       // olay kimliği
   const long&     lparam,   // long type event parameter
   const double&   dparam,   // long type event parameter
   const long&     sparam    // string type event parameter
   );
```

Parametreler

id

[in] ENUM_CHART_EVENT listesinden olay kimliği.

lparam

[in] long tipi olay parametresi

dparam

[in] double tipi olay parametresi

sparam

[in] string tipi olay parametresi

Geri dönüş değeri

Geri dönüş değeri yok

Not

Önceden tanımlanmış OnChartEvent() fonksiyonu kullanılarak yönetilebilecek 11 olay tipi vardır. Özel olaylar için, CHARTEVENT_CUSTOM ve CHARTEVENT_CUSTOM_LAST dahil olmak üzere 65535 kimlik vardır. Özel bir olay oluşturmak için, EventChartCustom() fonksiyonunu kullanın.

ENUM_CHART_EVENT listesinden kısa olay açıklamaları:

- CHARTEVENT_KEYDOWN — grafik penceresi odaktayken klavyede bir tuşa basma;
- CHARTEVENT_MOUSE_MOVE — fare ve fare tuşlarını hareket ettirme (eğer bir grafik için CHART_EVENT_MOUSE_MOVE=true ise);
- CHARTEVENT_OBJECT_CREATE — bir grafiksel nesne oluştur (eğer bir grafik için CHART_EVENT_OBJECT_CREATE=true ise);
- CHARTEVENT_OBJECT_CHANGE — özellikler iletişim kutusundan nesne özelliklerini değiştir;
- CHARTEVENT_OBJECT_DELETE — bir grafiksel nesneyi kaldır (eğer bir grafik için CHART_EVENT_OBJECT_DELETE=true ise);
- CHARTEVENT_CLICK — bir grafik üzerinde tıklama;
- CHARTEVENT_OBJECT_CLICK — bir grafiğe ait olan bir grafiksel nesneye fare tıklaması;
- CHARTEVENT_OBJECT_DRAG — fare ile bir grafiksel nesneyi sürükleme;
- CHARTEVENT_OBJECT_ENDEDIT — bir grafiksel nesnenin Edit girdi kutusunda metin düzenlemesini bitirme (OBJ_EDIT);
- CHARTEVENT_CHART_CHANGE — bir grafiği değiştir;

- CHARTEVENT_CUSTOM+n — n'nin 0 ile 65535 arasında olduğu özel olay kimliği. CHARTEVENT_CUSTOM_LAST son kabul edilebilir özel olay kimliğini içerir (CHARTEVENT_CUSTOM+65535).

Tüm MQL5 programları, uygulamanın ana iş parçacığı dışındaki diğer iş parçacıklarında çalışır. Ana uygulama iş parçacığı, tüm Windows sistem iletilerini ele almaktan sorumludur ve bu işlem sonucunda da kendi uygulaması için Windows iletileri üretir. Örneğin, fareyi bir grafikte hareket ettirme (WM_MOUSE_MOVE olayı), uygulama penceresinin sonraki görüntülenmesi için birkaç sistem iletisi oluşturur ve ayrıca grafik üzerinde çalıştırılan uzman danışmanlar ve göstergelere dahili iletiler gönderir. Ana uygulama iş parçacığının WM_PAINT sistem iletisini henüz işleme koymadığında (ve dolayısıyla değiştirilmiş grafiği henüz oluşturmadığında), bir uzman danışmanın veya bir göstergenin fare hareketi olayını halihazırda almış olduğu bir durum meydana gelebilir. Bu durumda, CHART_FIRST_VISIBLE_BAR grafik özelliği yalnızca grafik oluşturulduktan sonra değiştirilecek.

Her bir olay tipi için, OnChartEvent() fonksiyonunun girdileri ilgili olayı yönetmek için gerekli belirli değerlere sahiptir. Tablo, parametreler aracılığı ile aktarılan olayları ve değerleri listeler.

Olay	'id' parametre değeri	'lparam' parametre değeri	'dparam' parametre değeri	'sparam' parametre değeri
Tuşa basma olayı	CHARTEVENT_KEYDOWN	basılı tuş kodu	Anahtar basılı durumda tutulurken tuşa basma sayısı	Klavye tuşlarının durumunu açıklayan bit maskesinin dizi değeri
Fare olayları (eğer bir grafik için CHART_EVENT_MOUSE_MOVE=true ise)	CHARTEVENT_MOUSE_MOVE	X koordinatı	Y koordinatı	Fare tuşlarının durumunu açıklayan bit maskesinin dizi değeri
Fare tekerlek olayı (eğer bir grafik için CHART_EVENT_MOUSE_WHEEL=true ise)	CHARTEVENT_MOUSE_WHEEL	Tuşların ve fare düğmelerinin durum bayrakları, imlecin X ve Y koordinatları. Açıklamalara bakın; örnekteki	Fare tekerleğinin Delta değeri	—
Bir grafiksel nesne oluşturma (eğer bir graik için CHART_EVENT_OBJECT_CREATE =true ise)	CHARTEVENT_OBJECT_CREATE	—	—	Oluşturulan grafiksel nesnenin adı
Özellikler iletişim kutusu	CHARTEVENT_OBJECT_CHANGE	—	—	Değiştirilmiş grafiksel

Olay	'id' parametre değeri	'lparam' parametre değeri	'dparam' parametre değeri	'sparam' parametre değeri
aracılığı ile nesne özelliklerini değiştirme				nesnenin adı
Bir grafiksel nesneyi silme (eğer bir grafik için CHART_EVENT_OBJECT_DELETE =true ise)	CHARTEVENT_OBJECT_DELETE	—	—	Silinmiş grafiksel nesnenin adı
Bir grafikte fare tıklaması	CHARTEVENT_CLICK	X koordinatı	Y koordinatı	—
Bir grafiksel nesnede fare tıklaması	CHARTEVENT_OBJECT_CLICK	X koordinatı	Y koordinatı	Olayın meydana geldiği grafiksel nesnenin adı
Fare ile bir grafiksel nesneyi hareket ettirme	CHARTEVENT_OBJECT_DRAG	—	—	Hareket ettirilmiş grafiksel nesnenin adı
Bir "Girdi alanı" grafiksel nesnesinin girdi kutusunun metin düzenlemesini bitirme	CHARTEVENT_OBJECT_ENDEDIT	—	—	Metin düzenlemesinin bitirildiği "Girdi alanı" grafiksel nesnesinin ismi
Özellikler iletişim penceresi aracılığı ile grafiği yeniden boyutlandırma veya değiştirme	CHARTEVENT_CHART_CHANGE	—	—	—
N numaralı özel olay	CHARTEVENT_CUSTOM+N	EventChartCustom() fonksiyonu tarafından tanımlanan değer	EventChartCustom() fonksiyonu tarafından tanımlanan değer	EventChartCustom() fonksiyonu tarafından tanımlanan değer

Örnek grafik olay dinleyicisi:

```
//+------------------------------------------------------------------+
//|                                          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"
#property description "Örnek grafik olay dinleyicisi ve özel olay oluşturucusu"
//--- hizmet tuşları kimlikleri
#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
//+------------------------------------------------------------------+
//| Uzman danışman başlatımı                                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- CHARTEVENT_CUSTOM sabiti değerini görüntüle
   Print("CHARTEVENT_CUSTOM=",CHARTEVENT_CUSTOM);
//---
   Print("Uzman danışman çalıştırıldı ",MQLInfoString(MQL5_PROGRAM_NAME));
//--- grafik nesnesi oluşturma olaylarını almanın bayrağını ayarla
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_CREATE,true);
//--- grafik nesnesi kaldırma olaylarını almanın bayrağını ayarla
   ChartSetInteger(ChartID(),CHART_EVENT_OBJECT_DELETE,true);
//--- fare tekerleği kaydırma iletilerini etkinleştir
   ChartSetInteger(0,CHART_EVENT_MOUSE_WHEEL,1);
//--- grafik özelliklerinin zorla güncellenmesi olay işleme için hazırlık sağlar
   ChartRedraw();
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Uzman danışman tik fonksiyonu                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- özel bir olay oluşturmak için tik sayacı
   static int tick_counter=0;
//--- birikmiş tikleri bu değere bölün
   int simple_number=113;
//---
   tick_counter++;
```

```
//--- eğer tik sayacı simple_number ın katları ise özel bir olay gönder
   if(tick_counter%simple_number==0)
     {
      //--- 0'dan 65535'e kadar bir özel olay kimliği oluştur
      ushort custom_event_id=ushort(tick_counter%65535);
      //---  parametre doldurma ile özel bir olay gönder
      EventChartCustom(ChartID(),custom_event_id,tick_counter,SymbolInfoDouble(Symbol
      //--- örnek sonuçlarını analiz etmek için bir günlüğe ekle
      Print(__FUNCTION__,": Özel bir olay gönder, kimlik=",custom_event_id);
     }
//---
  }
//+------------------------------------------------------------------+
//| ChartEvent fonksiyonu                                            |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- tuş basımı
   if(id==CHARTEVENT_KEYDOWN)
     {
      switch((int)lparam)
        {
         case KEY_NUMLOCK_LEFT:  Print("KEY_NUMLOCK_LEFT e basıldı");   break;
         case KEY_LEFT:          Print("KEY_LEFT e basıldı");           break;
         case KEY_NUMLOCK_UP:    Print("KEY_NUMLOCK_UP a basıldı");     break;
         case KEY_UP:            Print("KEY_UP a basıldı");             break;
         case KEY_NUMLOCK_RIGHT: Print("KEY_NUMLOCK_RIGHT a basıldı");  break;
         case KEY_RIGHT:         Print("KEY_RIGHT a basıldı");          break;
         case KEY_NUMLOCK_DOWN:  Print("KEY_NUMLOCK_DOWN a basıldı");   break;
         case KEY_DOWN:          Print("KEY_DOWN a basıldı");           break;
         case KEY_NUMPAD_5:      Print("KEY_NUMPAD_5 a basıldı");       break;
         case KEY_NUMLOCK_5:     Print("KEY_NUMLOCK_5 a basıldı");      break;
         default:                Print("Listede olmayan tuşa basıldı");
        }
     }
//--- bir grafiğe sol tıklama
   if(id==CHARTEVENT_CLICK)
      Print("Grafik üzerindeki fare tıklama koordinatları: x = ",lparam,"  y = ",dpara
//--- bir grafiksel nesneye tıklama
   if(id==CHARTEVENT_OBJECT_CLICK)
      Print("Fare tıklaması yapıldı, nesne adı '"+sparam+"'");
//--- nesne kaldırıldı
   if(id==CHARTEVENT_OBJECT_DELETE)
      Print("Kaldırılan nesne adı ",sparam);
//--- nesne oluşturuldu
   if(id==CHARTEVENT_OBJECT_CREATE)
```

```
      Print("Oluşturulan nesne adı ",sparam);
//--- değiştirilen nesne
   if(id==CHARTEVENT_OBJECT_CHANGE)
      Print("Değiştirilen nesne adı ",sparam);
//--- nesne hareket ettirildi veya çapa noktalarının koordinatları değişti
   if(id==CHARTEVENT_OBJECT_DRAG)
      Print("Nesnenin çapa noktalarının değişimi, ilgili nesne adı ",sparam);
//--- Edit grafiksel nesnesinin girdi alanının metni değiştirildi
   if(id==CHARTEVENT_OBJECT_ENDEDIT)
      Print("Edit nesnesinde metin değişti, ",sparam,"  id=",id);
//--- fare hareket olayları
   if(id==CHARTEVENT_MOUSE_MOVE)
      Comment("POINT: ",(int)lparam,",",(int)dparam,"\n",MouseState((uint)sparam));
   if(id==CHARTEVENT_MOUSE_WHEEL)
     {
      //--- Bu olay için fare düğmelerinin ve tekerleğinin durumunu dikkate al
      int flg_keys = (int)(lparam>>32);          // Ctrl ve Shift tuşunun ve de fare
      int x_cursor = (int)(short)lparam;         // Fare tekerleği olayı meydana geldi
      int y_cursor = (int)(short)(lparam>>16);   // Fare tekerleği olayı meydana geldi
      int delta    = (int)dparam;                // +120 yada -120 ye ulaşılınca teti
      //--- bayrağı yönetme
      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=", tuşlar='"+StringSubstr(str_keys,0,StringLen(str_keys)-1)+"'";
      PrintFormat("%s: X=%d, Y=%d, delta=%d%s",EnumToString(CHARTEVENT_MOUSE_WHEEL),x_
     }
//--- özellikler iletişim penceresi kullanarak grafiği yeniden boyutlandırmanın veya
   if(id==CHARTEVENT_CHART_CHANGE)
      Print("grafik boyutunu ve özelliklerini değiştirme");
//--- özel olay
   if(id>CHARTEVENT_CUSTOM)
      PrintFormat("Özel olay, kimlik=%d, lparam=%d, dparam=%G, sparam=%s",id,lparam,dp
  }
//+------------------------------------------------------------------+
```

```
//| MouseState                                                        |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // fare sol
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // fare sağ
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // fare orta
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // fare ilk X tuşu
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // fare ikinci X tuşu
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // shift tuşu
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // control tuşu
   return(res);
  }
```

Ayrıca bakınız

EventChartCustom, Grafik olaylarının türleri, Olay yönetimi fonksiyonları, Program yürütme, Müşteri terminal olayları

OnTester

Fonksiyon, test işleminden sonra gerekli eylemleri gerçekleştirmek için Tester olayı meydana geldiğinde uzman danışmanlarda çağrılır.

```
double  OnTester(void);
```

Geri dönüş değeri

Test sonuçlarını değerlendirmek için özel kriter optimizasyonunun değeri

Not

OnTester() fonksiyonu, yalnızca uzman danışmanları test ederken kullanılabilir ve esasen girdi parametrelerini optimize ederken 'Custom max' kriteri olarak kullanılan bir değerin hesaplanması için tasarlanmıştır.

Genetik optimizasyon sırasında, bir nesil içerisindeki sonuçları sıralama azalan düzende gerçekleştirilir. Bu, en yüksek değere sahip sonuçların optimizasyon kriteri açısından en iyisi olduğu anlamına gelir. Bu tür sıralama için en kötü değerler sonda yerleşir ve devamında atılır. Dolayısıyla, gelecek neslin oluşturulmasında yer almazlar.

Böylece, OnTester() fonksiyonu, yalnızca kendi test sonuç raporlarınızı oluşturmanıza ve kaydetmenize olanak tanımaz, ayrıca alım-satım stratejisinin en iyi parametrelerini bulmak için optimizasyon sürecini kontrol etmenizi sağlar.

Aşağıda, özel kriter optimizasyonunun hesaplanması için bir **örnek** verilmiştir. Düşünce, denge grafiğinin doğrusal regresyonunu hesaplamaktır. Denge grafiği kullanılarak bir stratejinin optimize edilmesi ve "Balance + max Sharpe Ratio" kriteriyle sonuçların karşılaştırılması makalesinde anlatılmıştır.

```
//+------------------------------------------------------------------+
//|                                                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 "OnTester() yöneticisine sahip örnek uzman danışman"
#property description "Özel bir optimizasyon kriteri olarak, "
#property description "karesel sapma hatasına bölünen"
#property description "denge grafiği doğrusal regresyon oranı geri döndürülür"
//--- alım-satım operasyonları için sınıfı dahil et
#include <Trade\Trade.mqh>
//--- Uzman danışman girdi parametreleri
input double Lots                = 0.1;      // Hacim
input int    Slippage            = 10;       // İzin verilebilir kayma
input int    MovingPeriod        = 80;       // Hareket eden ortalama zaman aralığı
input int    MovingShift         = 6;        // Hareket eden ortalama kaydırma miktarı
//--- global değişkenler
int    IndicatorHandle=0;  // gösterge yönetimi
```

```
bool   IsHedging=false;    // hesabın bayrağı
CTrade trade;              // alım-satım işlemelerini gerçekleştirmek için
//---
#define EA_MAGIC 18052018
//+------------------------------------------------------------------+
//| Pozisyon açma durumunu kontrol et                                |
//+------------------------------------------------------------------+
void CheckForOpen(void)
  {
   MqlRates rt[2];
//--- sadece yeni bir barın başlangıcıyla işlem yap
   if(CopyRates(_Symbol,_Period,0,2,rt)!=2)
     {
      Print("CopyRates of ",_Symbol," başarısız oldu, geçmiş yok");
      return;
     }
//--- tik hacmi
   if(rt[1].tick_volume>1)
      return;
//--- hareketli ortalama değerlerini elde et
   double   ma[1];
   if(CopyBuffer(IndicatorHandle,0,1,1,ma)!=1)
     {
      Print("iMA'dan CopyBuffer başarısız oldu, veri yok");
      return;
     }
//--- sinyal varlığını kontrol et
   ENUM_ORDER_TYPE signal=WRONG_VALUE;
//--- mum daha yüksekten açıldı ancak hareketli ortalamanın altında kapandı
   if(rt[0].open>ma[0] && rt[0].close<ma[0])
      signal=ORDER_TYPE_BUY;    // alış sinyali
   else // mum daha düşükten açıldı ancak hareketli ortalamanın üzerinde kapandı
     {
      if(rt[0].open<ma[0] && rt[0].close>ma[0])
         signal=ORDER_TYPE_SELL;// satış sinyali
     }
//--- ek kontroller
   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);
        }
     }
//---
  }
//+------------------------------------------------------------------+
//| Pozisyon kapatma durumunu kontrol et                             |
```

```
//+------------------------------------------------------------------+
void CheckForClose(void)
  {
   MqlRates rt[2];
//--- sadece yeni bir barın başlangıcıyla işlem yap
   if(CopyRates(_Symbol,_Period,0,2,rt)!=2)
     {
      Print("CopyRates of ",_Symbol," başarısız oldu, geçmiş yok");
      return;
     }
   if(rt[1].tick_volume>1)
      return;
//--- hareketli ortalama değerlerini elde et
   double   ma[1];
   if(CopyBuffer(IndicatorHandle,0,1,1,ma)!=1)
     {
      Print("iMA'dan CopyBuffer başarısız oldu, veri yok");
      return;
     }
//--- pozisyon daha önceden PositionSelect() kullanılarak seçilmişti
   bool signal=false;
   long type=PositionGetInteger(POSITION_TYPE);
//--- mum daha yüksekten açıldı ancak hareketli ortalamanın altında kapandı - satış po
   if(type==(long)POSITION_TYPE_SELL && rt[0].open>ma[0] && rt[0].close<ma[0])
      signal=true;
//--- mum daha düşükten açıldı ancak hareketli ortalamanın üzerinde kapandı - alış poz
   if(type==(long)POSITION_TYPE_BUY && rt[0].open<ma[0] && rt[0].close>ma[0])
      signal=true;
//--- ek kontroller
   if(signal)
     {
      if(TerminalInfoInteger(TERMINAL_TRADE_ALLOWED) && Bars(_Symbol,_Period)>100)
         trade.PositionClose(_Symbol,Slippage);
     }
//---
  }
//+------------------------------------------------------------------+
//| Hesabın türünü göz önünde bulundurarak bir pozisyonu seç: Netting veya Hedging
//+------------------------------------------------------------------+
bool SelectPosition()
  {
   bool res=false;
//--- Hedging hesabı için bir pozisyon seç
   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;
           }
        }
     }
//--- Netting hesabı için bir pozisyon seç
   else
     {
      if(!PositionSelect(_Symbol))
         return(false);
      else
         return(PositionGetInteger(POSITION_MAGIC)==EA_MAGIC); //---Magix numarasını
     }
//--- gerçekleşim sonucu
   return(res);
  }
//+------------------------------------------------------------------+
//| Uzman danışman başlatma fonksiyonu                               |
//+------------------------------------------------------------------+
int OnInit(void)
  {
//--- alım-satım tipini seç: Netting yada Hedging
   IsHedging=((ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger(ACCOUNT_MARGIN_MODE)==ACCOU
//--- doğru pozisyon kontrolü için bir nesne başlatın
   trade.SetExpertMagicNumber(EA_MAGIC);
   trade.SetMarginMode();
   trade.SetTypeFillingBySymbol(Symbol());
   trade.SetDeviationInPoints(Slippage);
//--- Hareketli Ortalama göstergesini oluşturun
   IndicatorHandle=iMA(_Symbol,_Period,MovingPeriod,MovingShift,MODE_SMA,PRICE_CLOSE);
   if(IndicatorHandle==INVALID_HANDLE)
     {
      printf("iMA göstergesini oluştururken hata");
      return(INIT_FAILED);
     }
//--- tamam
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Uzman danışman tik fonksiyonu                                    |
//+------------------------------------------------------------------+
void OnTick(void)
  {
//--- eğer bir pozisyon halihazırda açıksa, kapatma durumunu kontrol et
   if(SelectPosition())
      CheckForClose();
// pozisyon açma durumunu kontrol et
```

```
   CheckForOpen();
//---
  }
//+------------------------------------------------------------------+
//| Tester fonksiyonu                                                |
//+------------------------------------------------------------------+
double OnTester()
  {
//--- özel kriter optimizasyon değeri (daha yüksek, daha iyidir)
   double ret=0.0;
//--- alım-satım işlem sonuçlarını diziye aktar
   double array[];
   double trades_volume;
   GetTradeResultsToArray(array,trades_volume);
   int trades=ArraySize(array);
//--- eğer 10'dan az işlem varsa, test pozitif sonuç vermez
   if(trades<10)
      return (0);
//--- işlem başına ortalama sonuç
   double average_pl=0;
   for(int i=0;i<ArraySize(array);i++)
      average_pl+=array[i];
   average_pl/=trades;
//--- tek-test modu için mesaj görüntüle
   if(MQLInfoInteger(MQL_TESTER) && !MQLInfoInteger(MQL_OPTIMIZATION))
      PrintFormat("%s: İşlemler=%d, Ortalama kar=%.2f",__FUNCTION__,trades,average_pl);
//--- kar grafiği için doğrusal regresyon oranlarını hesapla
   double a,b,std_error;
   double chart[];
   if(!CalculateLinearRegression(array,chart,a,b))
      return (0);
//--- regresyon hattından grafik sapmasının hatasını hesapla
   if(!CalculateStdError(chart,a,b,std_error))
      return (0);
//--- trend karlarının standart sapmaya oranını hesapla
   ret=(std_error == 0.0) ? a*trades : a*trades/std_error;
//--- özel kriter optimizasyon değerini geri döndür
   return(ret);
  }
//+------------------------------------------------------------------+
//| İşlemlerden karlar/zararların dizisini elde et                   |
//+------------------------------------------------------------------+
bool GetTradeResultsToArray(double &pl_results[],double &volume)
  {
//--- tüm işlem geçmişini talep et
   if(!HistorySelect(0,TimeCurrent()))
      return (false);
   uint total_deals=HistoryDealsTotal();
   volume=0;
```

```
//--- dizinin başlangıç boyutunu kenarlık ile ayarla - geçmişteki işlemlerin sayısına
   ArrayResize(pl_results,total_deals);
//--- işlem sonucunu düzelten, işlemlerin sayacı - kar yada zarar
   int counter=0;
   ulong ticket_history_deal=0;
//--- tüm işlemlerden geç
   for(uint i=0;i<total_deals;i++)
     {
      //--- bir işlem seç
      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
         //--- sadece alım-satım operasyonlarıyla ilgileniyoruz
         if((deal_type!=DEAL_TYPE_BUY) && (deal_type!=DEAL_TYPE_SELL))
            continue;
         //--- sadece karları/zararları düzelten işlemler
         if(deal_entry!=DEAL_ENTRY_IN)
           {
            //--- işlem sonuçlarını diziye yaz ve işlemlerin sayacını arttır
            pl_results[counter]=deal_profit;
            volume+=deal_volume;
            counter++;
           }
        }
     }
//--- dizinin son boyutunu ayarla
   ArrayResize(pl_results,counter);
   return (true);
  }
//+------------------------------------------------------------------+
//| Doğrusal regresyonu hesapla y=a*x+b                              |
//+------------------------------------------------------------------+
bool CalculateLinearRegression(double  &change[],double &chartline[],
                               double  &a_coef,double  &b_coef)
  {
//--- veri yeterliliğini kontrol et
   if(ArraySize(change)<3)
      return (false);
//--- bir birikim ile bir grafik dizisi oluştur
   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];
//--- şimdi, regresyon oranlarını hesapla
   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);
  }
//+------------------------------------------------------------------+
//|  Belirli a ve b için ortalama karesel sapma hatası               |
//+------------------------------------------------------------------+
bool  CalculateStdError(double  &data[],double  a_coef,double  b_coef,double &std_err)
  {
//--- karesel hataların toplamı
   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);
  }
```

Ayrıca bakınız

Alım-satım stratejilerini sınama, TesterHideIndicators, Optimizasyon sonuçlarıyla çalışma, TesterStatistics, OnTesterInit, OnTesterDeinit, OnTesterPass, MQL_TESTER, MQL_OPTIMIZATION, FileOpen, FileWrite, FileLoad, FileSave

OnTesterInit

Fonksiyon, strateji sınayıcısında optimizasyondan önce gerekli eylemleri gerçekleştirmek için TesterInit olayı meydana geldiğinde uzman danışmanlarda çağrılır. İki fonksiyon tipi vardır.

Sonucu geri göndüren versiyon

```
int  OnTesterInit(void);
```

Geri dönüş değeri

int tipi değeri; sıfır, optimizasyon başlamadan önce bir grafikte çalışmakta olan bir uzman danışmanın başarılı bir şekilde başlatılması anlamına gelir.

Gerçekleşim sonucunu geri döndüren OnTesterInit() çağrısı sadece program başlatmasına izin verdiği için değil, ayrıca optimizasyonun erken sonlandırılması durumunda hata kodunu geri döndürdüğünden dolayı kullanım için önerilmektedir. INIT_SUCCEEDED (0) dışındaki herhangi bir değerin geri döndürülmesi bir hata demektir, hiçbir optimizasyon başlatılmaz.

Sonucu geri döndüren versiyon sadece eski kodlarla uyumluluk için bırakılmıştır. Kullanım için önerilmemektedir

```
void  OnTesterInit(void);
```

Not

TesterInit olayı uzman danışmanın optimizasyonunun strateji sınayıcısında başlamasından önce oluşturulur. Bu olayda, OnTesterDeInit() veya OnTesterPass() olay yöneticilerine sahip bir uzman danışman otomatik olarak ayrı bir terminal grafiğine yüklenir. Strateji sınayıcısında belirtilen sembol ve zaman aralığına sahiptir.

Böyle bir olay, TesterInit, TesterDeinit ve TesterPass olaylarını alır, ancak Init, Deinit ve NewTick olaylarını almaz. Buna göre, optimizasyon sırasında her bir geçişin sonuçlarını işlemek için gerekli tüm mantık OnTesterInit (), OnTesterDeinit () ve OnTesterPass () yöneticilerinde uygulanmalıdır.

Bir strateji optimizasyonu sırasında her bir geçişin sonucu, FrameAdd () fonksiyonunu kullanan OnTester () yöneticisinden bir çerçeve aracılığıyla geçirilebilir.

OnTesterInit() fonksiyonu, optimizasyon sonuçlarının ileri işlemi için optimizasyonun başlangıcından önce bir uzman danışmanı başlatmak üzere tasarlanmıştır. Her zaman OnTesterDeinit() yöneticisiyle birlikte kullanılır.

OnTesterInit() gerçekleşim süresi sınırlıdır. Eğer aşılırsa, optimizasyonun kendisi iptal edilirken uzman danışman zorla durdurulur. Strateji sınayıcısı günlüğünde bir mesaj görüntülenir:

```
TesterOnTesterInit works too long. Tester cannot be initialized.
```

Örnek OnTick'den alınmıştır. Optimizasyon parametrelerini ayarlamak için OnTesterInit() yöneticisi eklenmiştir:

```
//+------------------------------------------------------------------+
//|                                        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"
#property description "Optimizasyon sırasında girdilerin değerlerini ve "
#property description "sınırlamalarını ayarlayan"
#property description "OnTesterInit() yöneticisine sahip örnek uzman danışman"

input double lots=0. 1;        // lot cinsinden hacim
input double kATR=3;           // ATR'de sinyal mum uzunluğu
input int    ATRperiod=20;     // ATR gösterge zaman aralığı
input int    holdbars=8;       // pozisyonu açık tutmak için gereken bar sayısı
input int    slippage=10;      // izin verilebilir kayma miktarı
input bool   revers=false;     // sinyali tersine döndür?
input ulong  EXPERT_MAGIC=0;   // Uzman danışmanın MagicNumber'ı
//--- ATR gösterge yönetimini depolamak için
int atr_handle;
//--- burada son ATR değerini ve mum bedenini saklayacağız
double last_atr,last_body;
datetime lastbar_timeopen;
double trade_lot;
//--- optimizasyon başlangıç zamanını hatırla
datetime optimization_start;
//--- optimizasyonun bitişinden sonra bir grafikte süreyi görüntülemek için
string report;
//+------------------------------------------------------------------+
//| TesterInit fonksiyonu                                            |
//+------------------------------------------------------------------+
void OnTesterInit()
  {
//--- optimizasyon için girdilerin değerlerini ayarla
   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("İlk değerler ve optimizasyon parametre sınırlamaları ayarlandı");
//--- optimizasyon başlangıç zamanını hatırla
   optimization_start=TimeLocal();
   report=StringFormat("%s: optimizasyon %s de başlatıldı",
                       __FUNCTION__,TimeToString(TimeLocal(),TIME_MINUTES|TIME_SECONDS
//--- mesajları grafikte ve terminal günlüğünde göster
   Print(report);
   Comment(report);
//---
  }
//+------------------------------------------------------------------+
//| TesterDeinit fonksiyonu                                          |
```

```
//+------------------------------------------------------------------+
void OnTesterDeinit()
  {
//--- optimizasyon süresi
   string log_message=StringFormat("%s: optimizasyon %d saniye sürdü",
                                   __FUNCTION__,TimeLocal()-optimization_start);
   PrintFormat(log_message);
   report=report+"\r\n"+log_message;
   Comment(report);
  }
//+------------------------------------------------------------------+
//| Uzman danışman başlatma fonksiyonu                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- global değişenleri başlat
   last_atr=0;
   last_body=0;
//--- doğru hacmi ayarla
   double min_lot=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
   trade_lot=lots>min_lot? lots:min_lot;
//--- ATR gösterge yöneticisini oluştur
   atr_handle=iATR(_Symbol,_Period,ATRperiod);
   if(atr_handle==INVALID_HANDLE)
     {
      PrintFormat("%s: iATR oluşturulamadı, hata kodu %d",__FUNCTION__,GetLastError())
      return(INIT_FAILED);
     }
//--- başarılı uzman danışman başlatma
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Uzman danışman tik fonksiyonu                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- alım-satım sinyali
   static int signal=0; // +1 alış sinyali anlamına gelmektedir, -1 satın sinyali anla
//--- 'holdbars' bar miktarından daha önce açılmış eski pozisyonları kontrol et ve kap
   ClosePositionsByBars(holdbars,slippage,EXPERT_MAGIC);
//--- yeni bir barı kontrol et
   if(isNewBar())
     {
      //--- sinyal varlığını kontrol et
      signal=CheckSignal();
     }
//--- eğer bir netleştirme pozisyonu açıksa, sinyali atla - kapanana kadar bekle
   if(signal!=0 && PositionsTotal()>0 && (ENUM_ACCOUNT_MARGIN_MODE)AccountInfoInteger
     {
```

```
      signal=0;
      return; // NewTick olay yöneticisinden çık ve yeni bir bar görünene kadar piyasa
     }
//--- bir hedging hesabı için, her bir pozisyon ayrı ayrı tutulur ve kapatılır
   if(signal!=0)
     {
      //--- alış sinyali
      if(signal>0)
        {
         PrintFormat("%s: Alış sinyali! Revers=%s",__FUNCTION__,string(revers));
         if(Buy(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
      //--- satış sinyali
      if(signal<0)
        {
         PrintFormat("%s: Satış sinyali! Revers=%s",__FUNCTION__,string(revers));
         if(Sell(trade_lot,slippage,EXPERT_MAGIC))
            signal=0;
        }
     }
//--- OnTick fonksiyon sonlandırma
  }
//+------------------------------------------------------------------+
//| Yeni bir alım-satım sinyali kontrol et                           |
//+------------------------------------------------------------------+
int CheckSignal()
  {
//--- 0, sinyal yok anlamına gelmektedir
   int res=0;
//--- sondan bir önceki tam barın ATR değerini elde et (bar indeksi 2'dir)
   double atr_value[1];
   if(CopyBuffer(atr_handle,0,2,1,atr_value)!=-1)
     {
      last_atr=atr_value[0];
      //--- son kapanan barın verilerini bir dizi MqlRates'e ilet
      MqlRates bar[1];
      if(CopyRates(_Symbol,_Period,1,1,bar)!=-1)
        {
         //--- son kapanan barın beden ölçüsünü hesapla
         last_body=bar[0].close-bar[0].open;
         //--- eğer son barın bedeni (index 1 olan bar) bir önceki ATR değerini aşarsa
         if(MathAbs(last_body)>kATR*last_atr)
            res=last_body>0?1:-1; // yükselen mum için pozitif değer
        }
      else
         PrintFormat("%s: Son bar elde edilemedi! Error",__FUNCTION__,GetLastError());
     }
   else
```

```
      PrintFormat("%s: ATR gösterge değeri alınamadı! Error",__FUNCTION__,GetLastError
//--- eğer ters alım-satım modu etkinse
   res=revers?-res:res;  // eğer gerekliyse sinyali tersine çevir (1 yerine -1'i geri
//--- bir alım-satım sinyal değerini geri döndür
   return (res);
  }
//+------------------------------------------------------------------+
//|  Yeni bir bar göründüğünde 'true'yu geri döndür                  |
//+------------------------------------------------------------------+
bool isNewBar(const bool print_log=true)
  {
   static datetime bartime=0; // mevcut barın açılış zamanını sakla
//--- sıfır indeksli barın açılış zamanını elde et
   datetime currbar_time=iTime(_Symbol,_Period,0);
//--- eğer açılış zaman bilgisi değişirse, yeni bir bar gelmiş demektir
   if(bartime!=currbar_time)
     {
      bartime=currbar_time;
      lastbar_timeopen=bartime;
      //--- yeni barın açılış zamanı verilerini günlükte göster
      if(print_log && !(MQLInfoInteger(MQL_OPTIMIZATION)||MQLInfoInteger(MQL_TESTER)))
        {
         //--- yeni barın açılış zamanı bilgisine sahip bir mesaj göster
         PrintFormat("%s: yeni bar %s %s üzerinde %s de açıldı",__FUNCTION__,_Symbol,
                     StringSubstr(EnumToString(_Period),7),
                     TimeToString(TimeCurrent(),TIME_SECONDS));
         //--- son fiyat(tik) hakkındaki verileri elde et
         MqlTick last_tick;
         if(!SymbolInfoTick(Symbol(),last_tick))
            Print("SymbolInfoTick() başarısız oldu, hata = ",GetLastError());
         //--- son fiyat zamanını milisaniye olarak kadar göster
         PrintFormat("Son fiyat %s.%03d taydı",
                     TimeToString(last_tick.time,TIME_SECONDS),last_tick.time_msc%1000
        }
      //--- yeni bir bar var
      return (true);
     }
//--- yeni bir bar yok
   return (false);
  }
//+------------------------------------------------------------------+
//| Belirtilen hacimle, piyasa fiyatından alış yap                   |
//+------------------------------------------------------------------+
bool Buy(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- piyasa fiyatından alış yap
   return (MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
```

```
//| Belirtilen hacimle, piyasa fiyatından satış yap                  |
//+------------------------------------------------------------------+
bool Sell(double volume,ulong deviation=10,ulong  magicnumber=0)
  {
//--- piyasa fiyatından alış yap
   return (MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber));
  }
//+------------------------------------------------------------------+
//| Bar sayısı ile hesaplanan açık tutma süresine göre pozisyonu kapat
//+------------------------------------------------------------------+
void ClosePositionsByBars(int holdtimebars,ulong deviation=10,ulong  magicnumber=0)
  {
   int total=PositionsTotal(); // açık pozisyonların sayısı
//--- tüm açık pozisyonları ara
   for(int i=total-1; i>=0; i--)
     {
      //--- pozisyon parametreleri
      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;

      //--- eğer bir pozisyonun ömrü, MagicNumber ve sembol eşleşirken, halihazırda fa
      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); // doğru mesaj formatı için metin kutusunu küçült
         PrintFormat(" #%d %s %s %.2f pozisyonunu kapat",
                     position_ticket,position_symbol,str_type,volume);
         //--- emir türünü belirle ve alım-satım isteğini gönder
         if(type==POSITION_TYPE_BUY)
            MarketOrder(ORDER_TYPE_SELL,volume,deviation,magicnumber,position_ticket);
         else
            MarketOrder(ORDER_TYPE_BUY,volume,deviation,magicnumber,position_ticket);
        }
     }
  }
//+------------------------------------------------------------------+
//| Bir alım-satım isteği hazırla ve gönder                          |
//+------------------------------------------------------------------+
bool MarketOrder(ENUM_ORDER_TYPE type,double volume,ulong slip,ulong magicnumber,ulong
  {
//--- yapıların bildirimi ve başlatımı
   MqlTradeRequest request={};
   MqlTradeResult  result={};
```

```
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(type==ORDER_TYPE_BUY)
      price=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- istek parametreleri
   request.action   =TRADE_ACTION_DEAL;                     // İşlem türü
   request.position =pos_ticket;                            // kapanıyorsa, pozisyon k
   request.symbol   =Symbol();                              // sembol
   request.volume   =volume;                                // hacim
   request.type     =type;                                  // emir türü
   request.price    =price;                                 // alım-satım fiyatı
   request.deviation=slip;                                  // fiyattan izin verilen s
   request.magic    =magicnumber;                           // emir MagicNumber'ı
//--- bir istek gönder
   if(!OrderSend(request,result))
     {
      //--- başarısızlık verilerini görüntüle
      PrintFormat("OrderSend %s %s %.2f %.5f de hata %d",
                  request.symbol,EnumToString(type),volume,request.price,GetLastError
      return (false);
     }
//--- başarılı operasyonu bildir
   PrintFormat("retcode=%u  işlem=%I64u  emir=%I64u",result.retcode,result.deal,result
   return (true);
  }
```

Ayrıca bakınız

Alım-satım stratejilerini sınama, Optimizasyon sonuçlarıyla çalışma, OnTesterDeinit, OnTesterPass, ParameterGetRange, ParameterSetRange

OnTesterDeinit

Fonksiyon, uzman danışman optimizasyonundan sonra TesterDeinit olayının meydana gelmesiyle uzman danışmanlarda çağrılır.

```
void  OnTesterDeinit(void);
```

Geri dönüş değeri

Geri dönüş değeri yok

Not

TesterDeinit olayı strateji sınayıcısındaki uzman danışman optimizasyonun bitiminden sonra oluşturulur.

OnTesterDeInit() veya OnTesterPass() olay yöneticisine sahip bir uzman danışman, optimizasyon başlangıcında ayrı bir grafikte otomatik olarak yüklenir. Strateji sınayıcısında belirtilmiş olan sembol ve zaman aralığına sahiptir. Fonksiyon, tüm optimizasyon sonuçlarının son işlenimi için tasarlanmıştır.

Test temsilcileri tarafından FrameAdd() fonksiyonunu kullanan test temsilcileri tarafından gönderilen optimizasyon çerçevelerinin deste halinde gelebileceğini ve teslimlerinin zaman alabileceğini unutmayın. Bu nedenle, TesterPass olaylarının yanı sıra çerçevelerin tümü, optimizasyonun bitiminden önce gelemez ve OnTesterPass()'ta işlenemez. Eğer OnTesterDeinit() deki tüm gecikmeli çerçeveleri almak istiyorsanız, FrameNext() fonksiyonunu kullanan kod bloğunu yerleştirin.

Ayrıca bakınız

Alım-satım stratejilerini sınama, Optimizasyon sonuçlarıyla çalışma, TesterStatistics, OnTesterInit, OnTesterPass, ParameterGetRange, ParameterSetRange

OnTesterPass

Fonksiyon, uzman danışman optimizasyonu sırasında yeni bir ver çerçevesini yönetmek için TesterPass olayının meydana gelmesiyle uzman danışmanlarda çağrılır.

```
void  OnTesterPass(void);
```

Geri dönüş değeri

Geri dönüş değeri yok

Not

TesterPass olayı, strateji sınayıcısındaki bir uzman danışman optimizasyonu esnasında bir çerçeve alınırken otomatik olarak oluşturulur.

OnTesterDeInit() veya OnTesterPass() olay yöneticisine sahip bir uzman danışman, optimizasyon başlangıcında ayrı bir grafikte otomatik olarak yüklenir. Strateji sınayıcısında belirtilmiş olan sembol ve zaman aralığına sahiptir. Fonksiyon, optimizasyon sırasında test temsilcilerinden alınan çerçeveleri yönetmek için tasarlanmıştır. Test sonuçlarını içeren çerçeve FrameAdd() fonksiyonunu kullanan OnTester()'dan gönderilmelidir.

FrameAdd()fonksiyonunu kullanan test temsilcileri tarafından gönderilen optimizasyon çerçevelerinin deste halinde gelebileceğini ve teslimlerinin zaman alabileceğini unutmayın. Bu nedenle, TesterPass olaylarının yanı sıra çerçevelerin tümü, optimizasyonun bitiminden önce gelemez ve OnTesterPass()'ta işlenemez. Eğer OnTesterDeinit() deki tüm gecikmeli çerçeveleri almak istiyorsanız, FrameNext() fonksiyonunu kullanan kod bloğunu yerleştirin.

OnTesterDeinit() optimizasyonu tamamlandıktan sonra, alınan tüm çerçeveleri FrameFirst()/FrameFilter ve FrameNext() fonksiyonlarını kullanarak yeniden sıralamak mümkündür.

Ayrıca bakınız

Alım-satım stratejilerini sınama, Optimizasyon sonuçlarıyla çalışma, OnTesterInit, OnTesterDeinit, FrameFirst, FrameFilter, FrameNext, FrameInputs

Piyasa Bilgisinin Alınması

Bunlar, piyasa durumu ile ilgili bilgi almak amacıyla tasarlanmış fonksiyonlardır.

Fonksiyon	Eylem
SymbolsTotal	Mevcut sembollerin sayısına (Piyasa Gözleminde seçilmiş olanlara veya hepsine) dönüş yapar
SymbolExist	Belirtilen ada sahip bir sembol olup olmadığını kontrol eder
SymbolName	Belirtilen sembolün ismine dönüş yapar
SymbolSelect	Piyasa Gözlemi penceresindeki bir sembolü seçer veya bir sembolü pencereden siler
SymbolIsSynchronized	Seçilen bir sembole dair terminalde bulunan veri ile alım satım sunucusundaki verinin senkronize olup olmadığını denetler
SymbolInfoDouble	Karşılık gelen özellik için, sembolün double tipli değerine dönüş yapar
SymbolInfoInteger	Karşılık gelen özellik için, belirtilen sembolün tam-sayı tipi (long, datetime, int veya bool) değerine dönüş yapar
SymbolInfoString	Karşılık gelen özellik için, belirtilen sembolün string tipli değerine dönüş yapar
SymbolInfoMarginRate	Returns the margin rates depending on the order type and direction
SymbolInfoTick	MqlTick tipi bir değişken ile, belirtilen sembol için mevcut fiyat değerlerine dönüş yapar
SymbolInfoSessionQuote	Belirtilen sembol ve gün için, belirtilen fiyatlama seanslarının başlangıç ve bitiş zamanlarının alınmasını sağlar.
SymbolInfoSessionTrade	Belirtilen sembol ve gün için, belirtilen alım-satım seanslarının başlangıç ve bitiş zamanlarının alınmasını sağlar.
MarketBookAdd	Seçilen bir sembol için Piyasa Derinliğinin açılmasını ve değişimler ile ilgili bilgi alınmasını sağlar
MarketBookRelease	Seçilen bir sembol için Piyasa Derinliğinin kapanmasını sağlayıp, değişimler ile ilgili bilgi alınmasını sonlandırır
MarketBookGet	Belirtilen bir sembolün Piyasa Derinliği kayıtlarını içeren MqlBookInfo yapı dizisine dönüş yapar

SymbolsTotal

Mevcut sembollerin sayısına (Piyasa Gözleminde seçilmiş olanlara veya hepsine) dönüş yapar.

```
int  SymbolsTotal(
   bool  selected      // True - sadece Piyasa Gözlemindeki semboller için
   );
```

Parametreler

selected

[in] İstek modu. 'true' veya 'false' olabilir.

Dönüş değeri

'selected' parametresinin 'true' değerini alması durumunda, Piyasa Gözlemi penceresinde seçili olan sembollerin sayısına dönüş yapar. Eğer parametre değeri 'false' ise, sembollerin toplam sayısına dönüş yapar.

SymbolExist

Belirtilen ada sahip bir sembol olup olmadığını kontrol eder.

```
bool  SymbolExist(
   const string  name,     // sembol adı
   bool&   is_custom       // kullanıcı tanımlı sembol adı
   );
```

Parametreler

name

[in] Sembol adı.

is_custom

[out] Başarılı yürütmenin akabinde ayarlanan kullanıcı tanımlı sembol özelliği. Eğer doğruysa, algılanan sembol bir kullanıcı tanımlı semboldür.

Geri dönüş değeri

Eğer yanlışsa, sembol standart ve kullanıcı tanımlı semboller arasında bulunmaz.

Ayrıca bakınız

SymbolsTotal, SymbolSelect, Kullanıcı tanımlı semboller

SymbolName

Sembol ismine dönüş yapar.

```
string  SymbolName(
   int   pos,           // listedeki numara
   bool  selected       // true - sadece Piyasa Gözlemindeki semboller için
   );
```

Parametreler

pos

[in] Sembolün listedeki sıra numarası.

selected

[in] İstek modu. Değer 'true' ise, sembol, Piyasa Gözleminde yer alan semboller listesinden alınır. Eğer değer 'false' ise, sembol genel listeden alınır.

Dönüş değeri

Sembol isminin string tipli değeri.

SymbolSelect

Piyasa Gözlemi penceresindeki bir sembolü seçer veya bir sembolü pencereden siler.

```
bool  SymbolSelect(
   string  name,        // sembol ismi
   bool    select       // ekle veya kaldır
   );
```

Parametreler

name

[in] Sembol ismi.

select

[in] Seçim. 'false' değeri seçilirse, Piyasa Gözlemi penceresinden bir sembol silinmelidir; aksi durumda ise, bu pencereden bir sembol seçilmelidir. Sembolün çizelge penceresi açıksa veya bu sembolde açık pozisyonlar varsa, sembol kaldırılamaz.

Dönüş değeri

Başarısızlık durumunda 'false' dönüşü yapar.

SymbolIsSynchronized

Seçilen bir sembole dair terminalde bulunan veri ile alım-satım sunucusundaki verinin senkronize olup olmadığını denetler

```
bool  SymbolIsSynchronized(
   string  name,        // sembol ismi
   );
```

Parametreler

name

[in] Sembol ismi.

Dönüş değeri

Veri senkronize ise 'true', aksi durumda 'false' dönüşü yapar.

Ayrıca Bakınız

SymbolInfoInteger, Veri Erişiminin Düzenlenmesi

SymbolInfoDouble

Belirtilen sembolün karşılık gelen özelliğine dönüş yapar. Fonksiyonun 2 çeşidi bulunmaktadır.

1. Hemen, özellik değerine dönüş yapar.

```
double  SymbolInfoDouble(
   string                    name,        // sembol
   ENUM_SYMBOL_INFO_DOUBLE   prop_id      // özellik tanımlayıcısı
   );
```

2. Fonksiyonun başarı durumuna göre 'true' veya 'false' değerine dönüş yapar. Başarı durumunda özelliğin değeri, son parametreye referansla geçirilen alıcı değişkene yerleştirilir.

```
bool  SymbolInfoDouble(
   string                    name,        // sembol
   ENUM_SYMBOL_INFO_DOUBLE   prop_id,     // özellik tanımlayıcı
   double&                   double_var   // burada özellik değerinin olduğunu farz edi
   );
```

Parametreler

name

[in] Sembol ismi.

prop_id

[in] Bir sembol özelliğinin tanımlayıcısı. Değer, ENUM_SYMBOL_INFO_DOUBLE sayımının değerlerinden biri olabilir.

double_var

[out] İstenen özelliği alması için double tipli değişken.

Dönüş değeri

double tipli değer. Çalışma hatası durumunda, hata bilgisi GetLastError() fonksiyonu ile elde edilebilir:

- 5040 - sembol ismi belirtmek için kullanılan string parametre hatalı,
- 4301 - bilinmeyen sembol (finansal enstrüman),
- 4302 - sembol "Piyasa Gözleminde" seçili değil (mevcut sembollerin listesinde bulunamadı),
- 4303 - sembol özelliği için geçersiz tanımlayıcı.

Not

Fonksiyon, son tik hakkında bilgi almak için kullanılacaksa SymbolInfoTick() fonksiyonunun kullanılması tavsiye edilir. Terminalin alım-satım hesabına bağlanmasından beri geçen sürede tek bir fiyat teklifi bile gelmemiş olabilir. Bu durumda, istenen değer tanımsız olacaktır.

Çoğu durumda, SymbolInfoTick() fonksiyonunun kullanımı, Alış fiyatı, Satış fiyatı, Son fiyat, hacim ve son tik zamanı gibi bilgilerin elde edilmesi için yeterlidir.

SymbolInfoMarginRate() fonksiyonu, emrin tipine ve yönüne bağlı olarak, alınan marjin miktarı ile ilgili veri sağlar.

Örnek:

```
void OnTick()
  {
//--- makas değerini sembol özelliklerinden alalım
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Spread %s = %I64d puan\r\n",
                            spreadfloat?"değişken":"sabit",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- şimdi, makas değerini kendimiz hesaplayalım
   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+"Hesaplanan makas = "+(string)spread_points+" puan";
   Comment(comm);
  }
```

SymbolInfoInteger

Belirtilen sembolün karşılık gelen özelliğine dönüş yapar. Fonksiyonun 2 çeşidi bulunmaktadır.

1. Hemen, özellik değerine dönüş yapar.

```
long  SymbolInfoInteger(
   string                     name,      // sembol
   ENUM_SYMBOL_INFO_INTEGER   prop_id    // özellik tanımlayıcısı

   );
```

2. Fonksiyonun başarı durumuna göre 'true' veya 'false' değerine dönüş yapar. Başarı durumunda özelliğin değeri, son parametreye referansla geçirilen alıcı değişkene yerleştirilir.

```
bool  SymbolInfoInteger(
   string                     name,      // sembol
   ENUM_SYMBOL_INFO_INTEGER   prop_id,   // özelliğin tanımlayıcısı
   long&                      long_var   // özelik değerinin burada olduğunu farz ediy
   );
```

Parametreler

name

[in] Sembol ismi.

prop_id

[in] Bir sembol özelliğinin tanımlayıcısı. Değer, ENUM_SYMBOL_INFO_INTEGER sayımının değerlerinden biri olabilir.

long_var

[out] İstenen özellik değerini alacak olan long tipli değişken.

Dönüş değeri

long tipli değer. Çalışma hatası durumunda, hata bilgisi GetLastError() fonksiyonu ile elde edilebilir:

- 5040 - sembol ismi belirtmek için kullanılan string parametre hatalı,
- 4301 - bilinmeyen sembol (finansal enstrüman),
- 4302 - sembol "Piyasa Gözleminde" seçili değil (mevcut sembollerin listesinde bulunamadı),
- 4303 - sembol özelliği için geçersiz tanımlayıcı.

Not

Fonksiyon, son tik hakkında bilgi almak için kullanılacaksa SymbolInfoTick() fonksiyonunun kullanılması tavsiye edilir. Terminalin alım-satım hesabına bağlanmasından beri geçen sürede tek bir fiyat teklifi bile gelmemiş olabilir. Bu durumda, istenen değer tanımsız olacaktır.

Çoğu durumda, SymbolInfoTick() fonksiyonunun kullanımı, Alış fiyatı, Satış fiyatı, Son fiyat, hacim ve son tik zamanı gibi bilgilerin elde edilmesi için yeterlidir.

Örnek:

```
void OnTick()
  {
```

```
//--- makas değerini sembol özelliklerinden alalım
   bool spreadfloat=SymbolInfoInteger(Symbol(),SYMBOL_SPREAD_FLOAT);
   string comm=StringFormat("Spread %s = %I64d puan\r\n",
                            spreadfloat?"değişken":"sabit",
                            SymbolInfoInteger(Symbol(),SYMBOL_SPREAD));
//--- şimdi, makas değerini kendimiz hesaplayalım
   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+"Hesaplanan makas = "+(string)spread_points+" puan";
   Comment(comm);
  }
```

SymbolInfoString

Belirtilen sembolün karşılık gelen özelliğine dönüş yapar. Fonksiyonun 2 çeşidi bulunmaktadır.

1. Hemen, özellik değerine dönüş yapar.

```
string  SymbolInfoString(
   string                    name,        // Sembol
   ENUM_SYMBOL_INFO_STRING   prop_id      // Özellik tanımlayıcı
   );
```

2. Fonksiyonun başarı durumuna göre, 'true' veya 'false' değerine dönüş yapar. Başarılı olması durumunda değer, son parametrede referans ile geçirilen değişkene atanır.

```
bool  SymbolInfoString(
   string                    name,        // Sembol
   ENUM_SYMBOL_INFO_STRING   prop_id,     // Özellik tanımlayıcı
   string&                   string_var   // Özelik değerini burada farz ediyoruz
   );
```

Parametreler

name

[in] Sembol ismi.

prop_id

[in] Bir sembol özelliğinin tanımlayıcısı. Bu değer, ENUM_SYMBOL_INFO_STRING sayımının değerlerinden biri olabilir.

string_var

[out] İstenen özellik değerini alacak olan string tipli değişken.

Dönüş değeri

string tipli değer. Çalışma hatası durumunda, hata bilgisi GetLastError() fonksiyonu ile elde edilebilir:

- 5040 - sembol ismi belirtmek için kullanılan string parametre hatalı,
- 4301 - bilinmeyen sembol (finansal enstrüman),
- 4302 - sembol "Piyasa Gözleminde" seçili değil (mevcut sembollerin listesinde bulunamadı),
- 4303 - sembol özelliği için geçersiz tanımlayıcı.

Not

Fonksiyon, son tik hakkında bilgi almak için kullanılacaksa SymbolInfoTick() fonksiyonunun kullanılması tavsiye edilir. Terminalin alım-satım hesabına bağlanmasından beri geçen sürede tek bir fiyat teklifi bile gelmemiş olabilir. Bu durumda, istenen değer tanımsız olacaktır.

Çoğu durumda, SymbolInfoTick() fonksiyonunun kullanımı, Alış fiyatı, Satış fiyatı, Son fiyat, hacim ve son tik zamanı gibi bilgilerin elde edilmesi için yeterlidir.

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
   );
```

Parametreler

name

[in] Sembol ismi.

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.

Dönüş Değeri

Returns true if request for properties is successful, otherwise false.

SymbolInfoTick

MqlTick tipi bir değişken ile belirtilen sembol için mevcut fiyat değerlerine dönüş yapar

```
bool  SymbolInfoTick(
   string    symbol,      // sembol ismi
   MqlTick&  tick         // bir yapıya yapılan referans
   );
```

Parametreler

symbol

[in] Sembol ismi.

tick

[out] Son fiyat güncellemesinin tarihinin ve mevcut fiyat değerinin yerleştirileceği MqlTick tipli yapıya verilen link.

Dönüş değeri

Fonksiyon, başarı durumunda 'true', aksi durumda 'false' dönüşü yapar.

SymbolInfoSessionQuote

Belirtilen sembol ve gün için, belirtilen fiyatlama seanslarının başlangıç ve bitiş zamanlarının alınmasını sağlar.

```
bool  SymbolInfoSessionQuote(
   string            name,                // sembol ismi
   ENUM_DAY_OF_WEEK  day_of_week,         // haftanın günü
   uint              session_index,       // seans indisi
   datetime&         from,                // seans başlangıç zamanı
   datetime&         to                   // seans bitiş zamanı
   );
```

Parametreler

name

[in] Sembol ismi.

ENUM_DAY_OF_WEEK

[in] Haftanın günü, ENUM_DAY_OF_WEEK değerlerinden biri olabilir.

uint

[in] Başlangıç ve bitiş zamanlarını almak istediğimiz seansın sıra numarası. Seansların indislenmesine 0 ile başlanır.

from

[out] 00 saat 00 dakikadan buyana geçmiş saniyeler bazında seans açılış zamanı. Dönüş değerinde tarih gözardı edilir.

to

[out] 00 saat 00 dakikadan buyana geçmiş saniyeler bazında seans kapanış zamanı. Dönüş değerinde tarih gözardı edilir.

Dönüş değeri

Belirtilen seans, sembol ve gün için istenen veriler alınmışsa 'true', aksi durunda 'false' dönüşü yapar.

Ayrıca Bakınız

Sembol Özellikleri, TimeToStruct, Veri Yapıları

SymbolInfoSessionTrade

Belirtilen sembol ve gün için, belirtilen alım-satım seanslarının başlangıç ve bitiş zamanlarının alınmasını sağlar.

```
bool  SymbolInfoSessionTrade(
   string            name,                // sembol ismi
   ENUM_DAY_OF_WEEK  day_of_week,         // haftanın günü
   uint              session_index,       // seans indisi
   datetime&         from,                // seans başlangıç zamanı
   datetime&         to                   // seans başlangıç zamanı
   );
```

Parametreler

name

[in] Sembol ismi.

ENUM_DAY_OF_WEEK

[in] Haftanın günü, ENUM_DAY_OF_WEEK değerlerinden biri olabilir.

uint

[in] Başlangıç ve bitiş zamanlarını almak istediğimiz seansın sıra numarası. Seansların indislenmesine 0 ile başlanır.

from

[out] 00 saat 00 dakikadan buyana geçmiş saniyeler bazında seans açılış zamanı. Dönüş değerinde tarih gözardı edilir.

to

[out] 00 saat 00 dakikadan buyana geçmiş saniyeler bazında seans kapanış zamanı. Dönüş değerinde tarih gözardı edilir.

Dönüş değeri

Belirtilen seans, sembol ve gün için istenen veriler alınmışsa 'true', aksi durunda 'false' dönüşü yapar.

Ayrıca Bakınız

Sembol Özellikleri, TimeToStruct, Veri Yapıları

MarketBookAdd

Seçilen bir sembol için Piyasa Derinliğinin açılmasını ve değişimler ile ilgili bilgi alınmasını sağlar

```
bool  MarketBookAdd(
   string  symbol      // sembol
   );
```

Parametreler

symbol

[in] Uzman Danışmanda veya script içinde, Piyasa Derinliğinin kullanılması istenen sembolün ismi

Dönüş değeri

Başarılı şekilde açılması durumunda 'true', aksi durumda 'false'.

Not

Normalde bu fonksiyon, OnInit() fonksiyonunun veya sınıf yapıcısının içinden çağrılmalıdır. Gelen uyarıların işlenmesi için Uzman Danışman, void OnBookEvent(string& symbol) fonksiyonunu içermelidir.

Ayrıca Bakınız

Piyasa Derinliğinin Yapısı, Yapılar ve Sınıflar

MarketBookRelease

Seçilen bir sembol için Piyasa Derinliğinin kapanmasını sağlayıp, değişimler ile ilgili bilgi alınmasını sonlandırır.

```
bool  MarketBookRelease(
   string  symbol      // sembol
   );
```

Parametreler

symbol

[in] Sembol ismi.

Dönüş değeri

Başarılı kapanma durumunda 'true', aksi durumda 'false' dönüşü yapar.

Not

MarketBookAdd() fonksiyonu, OnInit() fonksiyonundan çağrılmışsa, OnDeinit() fonksiyonundan da çağrılması gerekir. Veya benzer şekilde, MarketBookAdd() fonksiyonu bir sınıfın yapıcısından çağrılmışsa, sınıf yıkıcısı içinde de çağrılması gerekir.

Ayrıca Bakınız

Piyasa Derinliğinin Yapısı, Yapılar ve Sınıflar

MarketBookGet

Belirtilen sembolün Piyasa Derinliğinin kayıtlarını içeren, MqlBookInfo yapı dizisine dönüş yapar.

```
bool  MarketBookGet(
   string        symbol,     // sembol
   MqlBookInfo&  book[]      // bir diziye yapılan referans
   );
```

Parametreler

symbol

[in] Sembol ismi.

book[]

[in] Piyasa Derinliği kayıtlarından bir diziye yapılan referans. Dizi, yeterli sayıda kayıt alabilmek için önceden tahsis edilebilir. Eğer, bellek üzerinde daha önceden bir dinamik dizi tahsis edilmemişse, müşteri terminali tarafından tahsis edilir.

Dönüş değeri

Başarı durumunda 'true', aksi durumda 'false'.

Not

Piyasa Derinliği, MarketBookAdd() fonksiyonu ile önceden açılmalıdır.

Örnek:

```
   MqlBookInfo priceArray[];
   bool getBook=MarketBookGet(NULL,priceArray);
   if(getBook)
     {
      int size=ArraySize(priceArray);
      Print("MarketBookInfo yapısı",Symbol());
      for(int i=0;i<size;i++)
        {
         Print(i+":",priceArray[i].price
               +"    Volume = "+priceArray[i].volume,
               " type = ",priceArray[i].type);
        }
     }
   else
     {
      Print("sembolün Piyasa Derinliğinin içeriği alınamadı ",Symbol());
     }
```

Ayrıca Bakınız

Piyasa Derinliğinin Yapısı, Yapılar ve Sınıflar

Ekonomik takvim fonksiyonları

Bu bölümde, doğrudan MetaTrader platformunda bulunan ekonomik takvim ile çalışma adına fonksiyonlar açıklanmaktadır. Ekonomik takvim, makroekonomik göstergelerin açıklamalarını, yayımlama tarihlerini ve önem derecelerini içeren hazır bir ansiklopedidir. Makroekonomik göstergelerin ilgili değerleri, yayımlandığı anda MetaTrader platformuna gönderilir ve bu değerler, gerekli göstergeleri ülkeler, para birimleri ve önem açısından görsel olarak izlemenizi sağlayan etiketler halinde görüntülenir.

Ekonomik takvim ile çalışmak için tüm fonksiyonlar alım-satım sunucusu zamanını (TimeTradeServer) kullanır. Bu; MqlCalendarValue yapısındaki zamanın ve CalendarValueHistoryByEvent/CalendarValueHistory fonksiyonlarındaki zaman girdilerinin, kullanıcının yerel zamanından ziyade, alım-satım sunucusu zaman diliminde ayarlandığı anlamına gelir.

Ekonomik takvim fonksiyonları, gelen olayların otomatik olarak gerekli ülke/para birimleri perspektifinden özel önem kriterlerine göre analiz edilmesini sağlar.

Fonksiyon	Eylem
CalendarCountryById	ID'sine göre ülke açıklaması elde et
CalendarEventById	ID'sine göre olay açıklaması elde et
CalendarValueById	ID'sine göre olay değeri açıklaması elde et
CalendarCountries	Takvimde mevcut olan ülke adlarının dizisini elde et
CalendarEventByCountry	Takvimdeki tüm olayların açıklamalarını belirtilen ülke koduna göre elde et
CalendarEventByCurrency	Takvimdeki tüm olayların açıklamalarını belirtilen para birimine göre elde et
CalendarValueHistoryByEvent	Belirtilen zaman aralığındaki tüm olayların değer dizisini olay ID'sine göre elde et
CalendarValueHistory	Ülkeye ve/veya para birimine göre sıralan belirli bir zaman aralığındaki tüm olaylar için değer dizisini elde et
CalendarValueLastByEvent	Belirtilen bir change_id ile takvim veritabanı durumundan bu yana ID'sine göre olay değer dizisini elde et
CalendarValueLast	Belirtilen bir change_id ile takvim veritabanı durumundan bu yana ülkeye ve/veya para birimine göre sıralanan tüm etkinlikler için değer dizisini elde et

CalendarCountryById

ID'sine göre ülke açıklaması elde edin.

```
bool  CalendarCountryById(
   const long           country_id,     // ülke ID'si
   MqlCalendarCountry&  country         // ülke açıklaması almak için değişken
   );
```

Parametreler

country_id

[in] Ülke ID'si (ISO 3166-1).

country

[out] Ülke açıklaması almak için MqlCalendarCountry tip değişkeni

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata hakkında bilgi edinmek için, GetLastError() fonksiyonunu çağırın. Olası hatalar:

- 4001 - ERR_INTERNAL_ERROR (genel çalışma zamanı hatası),
- 5402 - ERR_CALENDAR_NO_DATA (ülke bulunamadı),
- 5401 - ERR_CALENDAR_TIMEOUT (istek süresi sınırı aşıldı).

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ekonomik takvimden ülkelerin listesini elde et
   MqlCalendarCountry countries[];
   int count=CalendarCountries(countries);
//--- sonucu kontrol et
   if(count==0)
      PrintFormat("CalendarCountries() 0 geri döndürdü! Error %d",GetLastError());
//--- eğer iki veya daha fazla ülke varsa
   if(count>=2)
     {
      MqlCalendarCountry country;
      //--- şimdi ID'sine göre ülke açıklaması elde et
      if(CalendarCountryById(countries[1].id, country))
        {
         //--- ülke açıklaması hazırla
         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);
         //--- ülke açıklamasını görüntüle
         Print(descr);
        }
      else
         Print("CalendarCountryById() başarısız oldu. Hata ",GetLastError());
     }
//---
  }
/*
  Sonuç:
   id = 999
   name = European Union
   code = EU
   currency = EUR
   currency_symbol = €
   url_name = european-union
*/
```

Ayrıca bakınız

CalendarCountries, CalendarEventByCountry

CalendarEventById

ID'sine göre olay açıklaması elde edin.

```
bool  CalendarEventById(
   ulong                event_id,      // olay ID'si
   MqlCalendarEvent&    event          // olay açıklaması almak için değişken
   );
```

Parametreler

event_id

[in] Olay ID'si

event

[out] Olay açıklaması almak için MqlCalendarEvent tip değişkeni.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata hakkında bilgi edinmek için, GetLastError() fonksiyonunu çağırın. Olası hatalar:

- 4001 - ERR_INTERNAL_ERROR (genel çalışma zamanı hatası),
- 5402 - ERR_CALENDAR_NO_DATA (ülke bulunamadı),
- 5401 - ERR_CALENDAR_TIMEOUT (istek süresi sınırı aşıldı).

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Almanya için ülke kodu (ISO 3166-1 Alpha-2)
   string germany_code="DE";
//--- Almanya olaylarını al
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(germany_code,events);
//--- Almanya olaylarını Günlük'te görüntüle
   if(events_count>0)
     {
      PrintFormat("Almanya olayları: %d",events_count);
      ArrayPrint(events);
     }
   else
     {
      PrintFormat("Şu ülke kodu için olaylar alınamadı: %s, hata %d",
                  germany_code,GetLastError());
      //--- komut dosyasının erken tamamlanması
      return;
     }
//--- events[] dizisindeki son olayın açıklamasını al
```

```
   MqlCalendarEvent event;
   ulong event_id=events[events_count-1].id;
   if(CalendarEventById(event_id,event))
     {
      MqlCalendarCountry country;
      CalendarCountryById(event.country_id,country);
      PrintFormat("event_id=%d ile olay açıklaması alındı",event_id);
      PrintFormat("Ülke: %s (ülke kodu = %d)",country.name,event.country_id);
      PrintFormat("Olay adı: %s",event.name);
      PrintFormat("Olay kodu: %s",event.event_code);
      PrintFormat("Olayın önemi: %s",EnumToString((ENUM_CALENDAR_EVENT_IMPORTANCE)ever
      PrintFormat("Olay tipi: %s",EnumToString((ENUM_CALENDAR_EVENT_TYPE)event.type));
      PrintFormat("Olay sektörü: %s",EnumToString((ENUM_CALENDAR_EVENT_SECTOR)event.se
      PrintFormat("Olay sıklığı: %s",EnumToString((ENUM_CALENDAR_EVENT_FREQUENCY)event
      PrintFormat("Olayın zamanı modu: %s",EnumToString((ENUM_CALENDAR_EVENT_TIMEMODE)
      PrintFormat("Olay ölçü birimi: %s",EnumToString((ENUM_CALENDAR_EVENT_UNIT)event.
      PrintFormat("Ondalık basamak sayısı: %d",event.digits);
      PrintFormat("Olay çarpanı: %s",EnumToString((ENUM_CALENDAR_EVENT_MULTIPLIER)ever
      PrintFormat("Kaynak URL: %s",event.source_url);
     }
   else
      PrintFormat("event_d=%s için olay açıklaması alma başarısız oldu, hata %d",
                  event_id,GetLastError());
  }
/*
  Sonuç:
  Almanya olayları: 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
  event_id=276500003 ile olay açıklaması alındı
  Ülke: Almanya (ülke kodu = 276)
  Olay adı: Markit Composite PMI
  Olay kodu: markit-composite-pmi
   Olay önemi: CALENDAR_IMPORTANCE_MODERATE
   Olay tipi: CALENDAR_TYPE_INDICATOR
   Olay sektörü: CALENDAR_SECTOR_BUSINESS
   Olay sıklığı: CALENDAR_FREQUENCY_MONTH
   Olayın zamanı modu: CALENDAR_TIMEMODE_DATETIME
   Olay ölçü birimi: CALENDAR_UNIT_NONE
  Ondalık basamak sayısı: 1
   Değer çarpanı: CALENDAR_MULTIPLIER_NONE
   Kaynak URL: https://www.markiteconomics.com
```

```
*/
```

Ayrıca bakınız

CalendarEventByCountry, CalendarEventByCurrency, CalendarValueById

CalendarValueById

ID'sine göre olay değeri açıklaması elde edin.

```
bool  CalendarValueById(
   ulong                value_id,      // olay değeri ID'si
   MqlCalendarValue&    value          // olay değeri almak için değişken
   );
```

Parametreler

value_id

[in] Olay değeri ID'si.

value

[out] Olay açıklaması almak için MqlCalendarValue tip değişkeni. Takvim olaylarını yönetme örneğine bakın.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata hakkında bilgi edinmek için, GetLastError() fonksiyonunu çağırın. Olası hatalar:

- 4001 - ERR_INTERNAL_ERROR (genel çalışma zamanı hatası),
- 5402 - ERR_CALENDAR_NO_DATA (ülke bulunamadı),
- 5401 - ERR_CALENDAR_TIMEOUT (istek süresi sınırı aşıldı).

Not

Ekonomik takvim ile çalışmak için tüm fonksiyonlar alım-satım sunucusu zamanını (TimeTradeServer) kullanır. Bu; MqlCalendarValue yapısındaki zamanın ve CalendarValueHistoryByEvent/CalendarValueHistory fonksiyonlarındaki zaman girdilerinin, kullanıcının yerel zamanından ziyade, alım-satım sunucusu zaman diliminde ayarlandığı anlamına gelir.

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Japonya için ülke kodu (ISO 3166-1 Alpha-2)
   string japan_code="JP";
//--- olayları aldığımız aralığın sınırlarını belirle
   datetime date_from=D'01.01.2018';  // 2018'deki tüm olayları al
   datetime date_to=0;                // 0, henüz gerçekleşmemiş olanlar dahil tüm bi
//--- Japonya olay değerleri dizisini al
   MqlCalendarValue values[];
   int values_count=CalendarValueHistory(values,date_from,date_to,japan_code);
//--- algılanan olay değerlerini gözden geçir
   if(values_count>0)
     {
```

```
      PrintFormat("Japonya olayları için değer sayısı: %d",values_count);
      //--- tüm "boş" değerleri sil (actual_value==-9223372036854775808)
      for(int i=values_count-1;i>=0;i--)
        {
         if(values[i].actual_value==-9223372036854775808)
            ArrayRemove(values,i,1);
        }
      PrintFormat("Boş olanları sildikten sonra değer sayısı: %d",ArraySize(values));
     }
   else
     {
      PrintFormat("Şu ülke kodu için olaylar alınamadı: %s, hata %d",
                  japan_code,GetLastError());
      //--- komut dosyasının erken tamamlanması
      return;
     }
//--- values[] dizisinde 10'dan fazla değer olmamasını sağla
   if(ArraySize(values)>10)
     {
      PrintFormat("Değer listesini 10'a düşür ve bu değerleri görüntüle");
      ArrayRemove(values,0,ArraySize(values)-10);
     }
   ArrayPrint(values);

//--- şimdi bilinen value_id ye dayanarak bir olay değeri açıklamasının nasıl elde edi
   for(int i=0;i<ArraySize(values);i++)
     {
      MqlCalendarValue value;
      CalendarValueById(values[i].id,value);
      PrintFormat("%d: value_id=%d değer=%d etki=%s",
                  i,values[i].id,value.actual_value,EnumToString(ENUM_CALENDAR_EVENT_
     }
//---
  }
/*
  Sonuç:
  Japonya olayları için değer sayısı: 1734
  Boş olanları sildikten sonra değer sayısı: 1017
  Değer listesini 10'a düşür ve bu değerleri görüntüle
        [id] [event_id]              [time]             [period] [revision] [actual_va
   [0] 56500  392030004 2019.03.28 23:30:00 2019.03.01 00:00:00          0         90
   [1] 56501  392030005 2019.03.28 23:30:00 2019.03.01 00:00:00          0         70
   [2] 56502  392030006 2019.03.28 23:30:00 2019.03.01 00:00:00          0        110
   [3] 56544  392030007 2019.03.28 23:30:00 2019.02.01 00:00:00          0        230
   [4] 56556  392050002 2019.03.28 23:30:00 2019.02.01 00:00:00          0        163
   [5] 55887  392020003 2019.03.28 23:50:00 2019.02.01 00:00:00          0         40
   [6] 55888  392020004 2019.03.28 23:50:00 2019.02.01 00:00:00          0       -180
   [7] 55889  392020002 2019.03.28 23:50:00 2019.02.01 00:00:00          0         20
   [8] 55948  392020006 2019.03.28 23:50:00 2019.02.01 00:00:00          1        140
```

```
   [9] 55949  392020007 2019.03.28 23:50:00 2019.02.01 00:00:00          1       -1000
  Value_id değerine dayanan olay değerlerine ilişkin kısa verileri görüntüle
   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
*/
```

Ayrıca bakınız

CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent, CalendarValueLast

CalendarCountries

Takvimde mevcut olan ülke adlarının dizisini elde edin.

```
int  CalendarCountries(
   MqlCalendarCountry&  countries[]         // Takvim ülkelerinin açıklamalarının bir
   );
```

Parametreler

countries[]

[out] Tüm Takvim ülkelerinin açıklamalarını almak için bir MqlCalendarCountry tip dizisi

Geri dönüş değeri

Alınan açıklama sayısı. Hata hakkında bilgi edinmek için, GetLastError() fonksiyonunu çağırın. Olası hatalar:

- 4001 - ERR_INTERNAL_ERROR (genel çalışma zamanı hatası),
- 5401 - ERR_CALENDAR_TIMEOUT (istek süresi sınırı aşıldı),
- 5400 - ERR_CALENDAR_MORE_DATA (dizi boyutu, tüm ülkelerin açıklamalarını almak için yetersiz; yalnızca diziye sığan açıklamalar alındı).

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ekonomik takvimden ülkelerin listesini elde et
   MqlCalendarCountry countries[];
   int count=CalendarCountries(countries);
//--- Günlükte diziyi görüntüle
   if(count>0)
      ArrayPrint(countries);
   else
      PrintFormat("CalendarCountries() 0 geri döndürdü! Error %d",GetLastError());
/*
  Sonuç:
        [id]            [name] [code] [currency] [currency_symbol]       [url_name] [re
   [ 0]    0 "Worldwide"      "WW"   "ALL"      ""                 "worldwide"
   [ 1]  999 "European Union" "EU"   "EUR"      "€"                "european-union"
   [ 2]  840 "United States"  "US"   "USD"      "$"                "united-states"
   [ 3]  124 "Canada"         "CA"   "CAD"      "$"                "canada"
   [ 4]   36 "Australia"      "AU"   "AUD"      "$"                "australia"
   [ 5]  554 "New Zealand"    "NZ"   "NZD"      "$"                "new-zealand"
   [ 6]  392 "Japan"          "JP"   "JPY"      "Ŧ"                "japan"
   [ 7]  156 "China"          "CN"   "CNY"      "Ŧ"                "china"
   [ 8]  826 "United Kingdom" "GB"   "GBP"      "J"                "united-kingdom"
   [ 9]  756 "Switzerland"    "CH"   "CHF"      "₣"                "switzerland"
```

```
   [10]  276 "Germany"        "DE"   "EUR"      "€"              "germany"
   [11]  250 "France"         "FR"   "EUR"      "€"              "france"
   [12]  380 "Italy"          "IT"   "EUR"      "€"              "italy"
   [13]  724 "Spain"          "ES"   "EUR"      "€"              "spain"
   [14]   76 "Brazil"         "BR"   "BRL"      "R$"             "brazil"
   [15]  410 "South Korea"    "KR"   "KRW"      "₩"              "south-korea"
*/
  }
```

Ayrıca bakınız

CalendarEventByCountry, CalendarCountryById

CalendarEventByCountry

Takvimdeki tüm olayların açıklamalarını belirtilen ülke koduna göre elde edin.

```
int  CalendarEventByCountry(
   string               country_code,     // kod olarak ülke adı (ISO 3166-1 alpha-2)
   MqlCalendarEvent&    events[]          // açıklama dizisi almak için değişken
   );
```

Parametreler

country_code

[in] Kod olarak ülke adı (ISO 3166-1 alpha-2)

events[]

[out] Belirtilen bir ülke için tüm olayların açıklamalarını almak için MqlCalendarEvent tip dizisi.

Geri dönüş değeri

Alınan açıklama sayısı. Hata hakkında bilgi edinmek için, GetLastError() fonksiyonunu çağırın. Olası hatalar:

- 4001 - ERR_INTERNAL_ERROR (genel çalışma zamanı hatası),
- 4004 - ERR_NOT_ENOUGH_MEMORY (istek yürütmek için yeterli bellek yok),
- 5401 - ERR_CALENDAR_TIMEOUT (istek süresi sınırı aşıldı),
- ArrayResize()'ın başarısız yürütme hataları

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- AB için ülke kodu (ISO 3166-1 Alpha-2)
   string EU_code="EU";
//--- AB olaylarını al
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(EU_code,events);
//--- AB olaylarını Günlük'te görüntüle
   if(events_count>0)
     {
      PrintFormat("AB olayları: %d",events_count);
      ArrayPrint(events);
     }
//---
  }
/*
  Sonuç:
  AB olayları: 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
   ...

*/
```

Ayrıca bakınız

CalendarCountries, CalendarCountryById

CalendarEventByCurrency

Takvimdeki tüm olayların açıklamalarını belirtilen para birimine göre elde edin.

```
int  CalendarEventByCurrency(
   const string         currency,     // kod olarak ülke para birimi
   MqlCalendarEvent&    events[]      // açıklama dizisi almak için değişken
   );
```

Parametreler

currency

[in] Kod olarak ülke para birimi.

events[]

[out] Belirtilen bir para birimi için tüm olayların açıklamalarını almak için MqlCalendarEvent tip dizisi.

Geri dönüş değeri

Alınan açıklama sayısı. Hata hakkında bilgi edinmek için, GetLastError() fonksiyonunu çağırın. Olası hatalar:

- 4001 - ERR_INTERNAL_ERROR (genel çalışma zamanı hatası),
- 4004 - ERR_NOT_ENOUGH_MEMORY (istek yürütmek için yeterli bellek yok),
- 5401 - ERR_CALENDAR_TIMEOUT (istek süresi sınırı aşıldı),
- ArrayResize()'ın başarısız yürütme hataları

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- ekonomik takvim olaylarını almak için dizi bildir
      MqlCalendarEvent events[];
//--- AB para birimi olaylarını al
      int count = CalendarEventByCurrency("EUR",events);
      Print("count = ", count);
//--- Mevcut örnek için 10 olay yeterlidir
      if(count>10)
         ArrayResize(events,10);
//--- olayları Günlük'te görüntüle
      ArrayPrint(events);
  }
/*
  Sonuç:
             [id] [type] [country_id] [unit] [importance]
   [0] 999010001      0          999      0            2 "https://www.ecb.europa.eu/ho
   [1] 999010002      0          999      0            2 "https://www.ecb.europa.eu/ho
```

```
   [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
*/
```

Ayrıca bakınız

CalendarEventById, CalendarEventByCountry

CalendarValueHistoryByEvent

Belirtilen zaman aralığındaki tüm olayların değer dizisini olay ID'sine göre elde edin.

```
bool  CalendarValueHistoryByEvent(
   ulong              event_id,          // olay ID'si
   MqlCalendarValue&  values[],          // değer açıklamaları için dizi
   datetime           datetime_from,     // bir zaman aralığının sol sınırı
   datetime           datetime_to=0      // bir zaman aralığının sağ sınırı
   );
```

Parametreler

event_id

[in] Olay ID'si

values[]

[out] Olay değerlerini almak için MqlCalendarValue tip dizisi. Takvim olaylarını yönetme örneğine bakın.

datetime_from

[in] Olayların belirtilen bir ID tarafından seçildiği zaman aralığının başlangıç tarihi (*datetime_from* < *datetime_to* durumundayken).

datetime_to=0

[in] Olayların belirtilen bir ID tarafından seçildiği zaman aralığının bitiş tarihi. Eğer *datetime_to* ayarlanmadıysa (veya 0 ise), Takvim veritabanındaki belirtilen *datetime_from* tarihinden başlayarak tüm olay değerleri (gelecekteki olayların değerleri dahil) geri döndürülür.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata hakkında bilgi edinmek için, GetLastError() fonksiyonunu çağırın. Olası hatalar:

- 4001 - ERR_INTERNAL_ERROR (genel çalışma zamanı hatası),
- 4004 - ERR_NOT_ENOUGH_MEMORY (istek yürütmek için yeterli bellek yok),
- 5401 - ERR_CALENDAR_TIMEOUT (istek süresi sınırı aşıldı),
- 5400 - ERR_CALENDAR_MORE_DATA (dizi boyutu, tüm değerlerin açıklamalarını almak için yetersiz; yalnızca diziye sığan açıklamalar alındı),
- ArrayResize()'ın başarısız yürütme hataları

Not

Ekonomik takvim ile çalışmak için tüm fonksiyonlar alım-satım sunucusu zamanını (TimeTradeServer) kullanır. Bu; MqlCalendarValue yapısındaki zamanın ve CalendarValueHistoryByEvent/CalendarValueHistory fonksiyonlarındaki zaman girdilerinin, kullanıcının yerel zamanından ziyade, alım-satım sunucusu zaman diliminde ayarlandığı anlamına gelir.

MqlCalendarValue yapısı, actual_value, forecast_value, prev_value ve revised_prev_value alanlarındaki değerleri kontrol etmek ve ayarlamak için yöntemler sağlar. Değer belirlenmezse, alan LONG_MIN (-9223372036854775808) olarak ayarlanır.

Lütfen bu alanlarda bulunan değerlerin bir milyon ile çarpıldığını unutmayın. Bunun anlamı, CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent ve CalendarValueLast fonksiyonlarını kullanarak MqlCalendarValue'da değerler aldığınızda, alandaki değerlerin LONG_MIN'e eşit olup olmadığını kontrol etmeniz; alanda bir değer belirlenmişse, değeri elde etmek için değeri 1.000.000'a bölmeniz gerektiği anlamına gelir. Değerleri elde etmenin diğer bir yöntemi de MqlCalendarValue yapısının fonksiyonlarını kullanarak değerleri kontrol etmek ve elde etmektir.

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- AB için ülke kodu (ISO 3166-1 Alpha-2)
   string EU_code="EU";
//--- AB olaylarını al
   MqlCalendarEvent events[];
   int events_count=CalendarEventByCountry(EU_code,events);
//--- AB olaylarını Günlük'te görüntüle
   if(events_count>0)
     {
      PrintFormat("AB olayları: %d",events_count);
      //--- olay listesini azalt, analiz için 10 olay yeterlidir
      ArrayResize(events,10);
      ArrayPrint(events);
     }
//--- "ECB Interest Rate Decision" olayının event_id=999010007 ye sahip olduğunu görür
   ulong event_id=events[6].id;        // olayın ID'si Takvim'de değişebilir, bu yüzde
   string event_name=events[6].name;   // Takvim olayının adı
   PrintFormat("event_name=%s event_id=%d için değerleri elde et",event_name,event_id)
//--- "ECB Interest Rate Decision" olayının tüm değerlerini al
   MqlCalendarValue values[];
//--- olayları aldığımız aralığın sınırlarını belirle
   datetime date_from=0;           // tüm olayları kullanılabilir geçmişin başlangıcır
   datetime date_to=D'01.01.2016'; // 2016'dan eski olmayan olayları al
   if(CalendarValueHistoryByEvent(event_id,values,date_from,date_to))
     {
      PrintFormat("%s için değerler alındı: %d",
                  event_name,ArraySize(values));
      //--- değer listesini azalt, analiz için 10 olay yeterlidir
      ArrayResize(values,10);
      ArrayPrint(values);
     }
   else
     {
      PrintFormat("Hata! event_id=%d için değerler alınamadı",event_id);
      PrintFormat("Hata kodu: %d",GetLastError());
     }
```

```
  }
//---
/*
  Sonuç:
  AB olayları: 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
  event_name=ECB Interest Rate Decision event_id=999010007 için değerleri elde et
  ECB Interest Rate Decision olay değerleri alındı: 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(
*/
```

Ayrıca bakınız

CalendarCountries, CalendarEventByCountry, CalendarValueHistory, CalendarEventById, CalendarValueById

CalendarValueHistory

Ülkeye ve/veya para birimine göre sıralan belirli bir zaman aralığındaki tüm olaylar için değer dizisini elde edin.

```
bool  CalendarValueHistory(
   MqlCalendarValue&  values[],              // değer açıklamaları için dizi
   datetime           datetime_from,         // bir zaman aralığının sol sınırı
   datetime           datetime_to=0          // bir zaman aralığının sağ sınırı
   const string       country_code=NULL,     // kod olarak ülke adı (ISO 3166-1 alpha-
   const string       currency=NULL          // kod olarak ülke para birimi
   );
```

Parametreler

values[]

[out] Olay değerlerini almak için MqlCalendarValue tip dizisi. Takvim olaylarını yönetme örneğine bakın.

datetime_from

[in] Olayların belirtilen bir ID tarafından seçildiği zaman aralığının başlangıç tarihi (*datetime_from* < *datetime_to* durumundayken).

datetime_to=0

[in] Olayların belirtilen bir ID tarafından seçildiği zaman aralığının bitiş tarihi. Eğer *datetime_to* ayarlanmadıysa (veya 0 ise), Takvim veritabanındaki belirtilen *datetime_from* tarihinden başlayarak tüm olay değerleri (gelecekteki olayların değerleri dahil) geri döndürülür.

country_code=NULL

[in] Kod olarak ülke adı (ISO 3166-1 alpha-2)

currency=NULL

[in] Kod olarak ülke para birimi.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata hakkında bilgi edinmek için, GetLastError() fonksiyonunu çağırın. Olası hatalar:

- 4001 - ERR_INTERNAL_ERROR (genel çalışma zamanı hatası),
- 4004 - ERR_NOT_ENOUGH_MEMORY (istek yürütmek için yeterli bellek yok),
- 5401 - ERR_CALENDAR_TIMEOUT (istek süresi sınırı aşıldı),
- 5400 - ERR_CALENDAR_MORE_DATA (dizi boyutu, tüm değerlerin açıklamalarını almak için yetersiz; yalnızca diziye sığan açıklamalar alındı),
- ArrayResize()'ın başarısız yürütme hataları

Not

Ekonomik takvim ile çalışmak için tüm fonksiyonlar alım-satım sunucusu zamanını (TimeTradeServer) kullanır. Bu; MqlCalendarValue yapısındaki zamanın ve CalendarValueHistoryByEvent/CalendarValueHistory fonksiyonlarındaki zaman girdilerinin, kullanıcının yerel zamanından ziyade, alım-satım sunucusu zaman diliminde ayarlandığı anlamına gelir.

Eğer *events[]* sabit uzunluklu dizi fonksiyona iletildiyse ve sonucun tamamını kaydetmek için yeterli alan yoksa, ERR_CALENDAR_MORE_DATA (5400) hatası etkinleştirilir.

Eğer *datetime_to* ayarlanmadıysa (veya 0 ise), Takvim veritabanındaki belirtilen *datetime_from* tarihinden başlayarak tüm olay değerleri (gelecekteki olayların değerleri dahil) geri döndürülür.

country_code ve *currency* filtreleri için NULL ve "" değerleri eşdeğerdir ve filtrenin yokluğu anlamına gelir.

country_code için, MqlCalendarCountry yapısının *kod* alanı, örneğin "US", "RU" veya "EU" kullanılmalıdır.

currency için, MqlCalendarCountry yapısının *currency* alanı, örneğin "USD", "RUB" veya "EUR" kullanılmalıdır.

Filtreler bir arada uygulanır, yani mantıksal 'VE' sadece her iki koşulun (ülke ve para birimi) aynı anda karşılandığı olayların değerlerini seçmek için kullanılır.

MqlCalendarValue yapısı, actual_value, forecast_value, prev_value ve revised_prev_value alanlarındaki değerleri kontrol etmek ve ayarlamak için yöntemler sağlar. Değer belirlenmezse, alan LONG_MIN (-9223372036854775808) olarak ayarlanır.

Lütfen bu alanlarda bulunan değerlerin bir milyon ile çarpıldığını unutmayın. Bunun anlamı, CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent ve CalendarValueLast fonksiyonlarını kullanarak MqlCalendarValue'da değerler aldığınızda, alandaki değerlerin LONG_MIN'e eşit olup olmadığını kontrol etmeniz; alanda bir değer belirlenmişse, değeri elde etmek için değeri 1.000.000'a bölmeniz gerektiği anlamına gelir. Değerleri elde etmenin diğer bir yöntemi de MqlCalendarValue yapısının fonksiyonlarını kullanarak değerleri kontrol etmek ve elde etmektir.

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- AB için ülke kodu (ISO 3166-1 Alpha-2)
   string EU_code="EU";
//--- tüm AB olay değerlerini al
   MqlCalendarValue values[];
//--- olayları aldığımız aralığın sınırlarını belirle
   datetime date_from=D'01.01.2018';  // 2018'deki tüm olayları al
   datetime date_to=0;                // 0, henüz gerçekleşmemiş olanlar dahil tüm bi
//--- 2018 yılından bu yana AB olay geçmişini talep et
   if(CalendarValueHistory(values,date_from,date_to,EU_code))
     {
      PrintFormat("country_code=%s ülke kodu için olay değerleri alındı: %d",
                  EU_code,ArraySize(values));
      //--- Günlük çıktısı için dizinin boyutunu küçült
      ArrayResize(values,10);
//--- olay değerlerini Günlük'te görüntüle
      ArrayPrint(values);
```

```
      }
    else
      {
       PrintFormat("Hata! Şu ülke kodu için olaylar alınamadı: country_code=%s",EU_code
       PrintFormat("Hata kodu: %d",GetLastError());
      }
//---
  }
/*
  Sonuç:
  country_code=EU ülke kodu için olay değerleri alındı: 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
*/
```

Ayrıca bakınız

CalendarCountries, CalendarEventByCountry, CalendarValueHistoryByEvent, CalendarEventById, CalendarValueById

CalendarValueLastByEvent

Belirtilen bir change_id ile takvim veritabanı durumundan bu yana ID'sine göre olay değer dizisini elde edin.

```
int  CalendarValueLastByEvent(
   ulong                event_id,      // olay ID'si
   ulong&               change_id,     // Takvim change ID'si
   MqlCalendarValue&    values[]       // değer açıklamaları için dizi
   );
```

Parametreler

event_id

[in] Olay ID'si

change_id

[in][out] Change ID.

values[]

[out] Olay değerlerini almak için MqlCalendarValue tip dizisi. Takvim olaylarını yönetme örneğine bakın.

Geri dönüş değeri

Alınan olay değeri sayısı. Hata hakkında bilgi edinmek için, GetLastError() fonksiyonunu çağırın. Olası hatalar:

- 4001 - ERR_INTERNAL_ERROR (genel çalışma zamanı hatası),
- 4004 - ERR_NOT_ENOUGH_MEMORY (istek yürütmek için yeterli bellek yok),
- 5401 - ERR_CALENDAR_TIMEOUT (istek süresi sınırı aşıldı),
- 5400 - ERR_CALENDAR_MORE_DATA (dizi boyutu, tüm değerlerin açıklamalarını almak için yetersiz; yalnızca diziye sığan açıklamalar alındı),
- ArrayResize()'ın başarısız yürütme hataları

Not

Ekonomik takvim ile çalışmak için tüm fonksiyonlar alım-satım sunucusu zamanını (TimeTradeServer) kullanır. Bu; MqlCalendarValue yapısındaki zamanın ve CalendarValueHistoryByEvent/CalendarValueHistory fonksiyonlarındaki zaman girdilerinin, kullanıcının yerel zamanından ziyade, alım-satım sunucusu zaman diliminde ayarlandığı anlamına gelir.

Eğer *events[]* sabit uzunluklu dizi fonksiyona iletildiyse ve sonucun tamamını kaydetmek için yeterli alan yoksa, ERR_CALENDAR_MORE_DATA (5400) hatası etkinleştirilir.

Eğer *change_id* = 0 fonksiyona iletilirse, fonksiyon her zaman sıfır geri döndürür, ancak geçerli takvim veritabanı *change_id* halinde geri döndürülür.

Fonksiyon, belirli bir haber için diziyi ve haberin yeni değerlerini almak adına fonksiyonun sonraki çağrıları için kullanılabilecek yeni bir *change_id* yi geri döndürür. Böylece, bu fonksiyonu bilinen en son *change_id* ile çağırarak belirli bir habere ait değerleri güncellemek mümkündür.

MqlCalendarValue yapısı, actual_value, forecast_value, prev_value ve revised_prev_value alanlarındaki değerleri kontrol etmek ve ayarlamak için yöntemler sağlar. Değer belirlenmezse, alan LONG_MIN (-9223372036854775808) olarak ayarlanır.

Lütfen bu alanlarda bulunan değerlerin bir milyon ile çarpıldığını unutmayın. Bunun anlamı, CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent ve CalendarValueLast fonksiyonlarını kullanarak MqlCalendarValue'da değerler aldığınızda, alandaki değerlerin LONG_MIN'e eşit olup olmadığını kontrol etmeniz; alanda bir değer belirlenmişse, değeri elde etmek için değeri 1.000.000'a bölmeniz gerektiği anlamına gelir. Değerleri elde etmenin diğer bir yöntemi de MqlCalendarValue yapısının fonksiyonlarını kullanarak değerleri kontrol etmek ve elde etmektir.

Nonfarm payrolls olayını dinleyen örnek Uzman Danışman:

```
#property description "Nonfarm Payrolls olayını izlemek adına"
#property description " CalendarValueLastByEvent fonksiyonunu kullanım örneği."
#property description "Bunu başarmak için Takvim veritabanının"
#property description " mevcut change ID'sini elde edin. Devamında, zamanlayıcı anketi
#property description " almak için bu ID'yi kullanın"
//+------------------------------------------------------------------+
//| Uzman danışman başlatma fonksiyonu                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- zamanlayıcı oluştur
   EventSetTimer(60);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Uzman danışman sonlandırma fonksiyonu                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- zamanlayıcıyı kaldır
   EventKillTimer();
  }
//+------------------------------------------------------------------+
//| Uzman danışman tik fonksiyonu                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Zamanlayıcı fonksiyonu                                           |
//+------------------------------------------------------------------+
void OnTimer()
  {
```

```
//--- Takvim veritabanı change ID'si
   static ulong calendar_change_id=0;
//--- ilk çalıştırmanın işareti
   static bool first=true;
//--- olay ID'si
   static ulong event_id=0;
//--- olay adı
   static string event_name=NULL;
//--- olay değeri dizisi
   MqlCalendarValue values[];
//--- başlatmayı gerçekleştir - mevcut calendar_change_id yi elde et
   if(first)
     {
      MqlCalendarEvent events[];
      //--- ABD için ülke kodu (ISO 3166-1 Alpha-2)
      string USA_code="US";
      //--- ABD olaylarını al
      int events_count=CalendarEventByCountry(USA_code,events);
      //--- 'events' dizisinde gerekli bir olayın konumu
      int event_pos=-1;
      //--- Günlükte ABD olaylarını görüntüle
      if(events_count>0)
        {
         PrintFormat("%s: ABD olayları: %d",__FUNCTION__,events_count);
         for(int i=0;i<events_count;i++)
           {
            string event_name_low=events[i].name;
            //--- olay adını küçük harf haline değiştir
            if(!StringToLower(event_name_low))
              {
               PrintFormat("StringToLower() %d hatasını geri döndürdü",GetLastError())
               //--- vaktinden önce fonksiyondan çık
               return;
              }
            //--- "Nonfarm Payrolls" olayı için arama yap
            if(StringFind(event_name_low,"nonfarm payrolls")!=-1)
              {
               //--- olay bulundu, bu ID'yi hatırla
               event_id=events[i].id;
               //--- "Nonfarm Payrolls" olay adını yaz
               event_name=events[i].name;
               //--- 'events[]' dizisindeki olayların konumunu hatırla
               event_pos=i;
               //--- Takvimin, adlarında "nonfarm payrolls" içeren birkaç olaya sahip
               PrintFormat("\"Nonfarm Payrolls\" olayı bulundu: event_id=%d  event_nar
               //--- bu örneği daha iyi anlamak için 'break' operatörünü derleme dışı
               break;
              }
           }
```

```
         //--- "Nonfarm Payrolls"dan sonraki olayları silerek listeyi azaltın
         ArrayRemove(events,event_pos+1);
         //--- daha uygun analiz için "Nonfarm Payrolls"dan önce 9 olay bırak
         ArrayRemove(events,0,event_pos-9);
         ArrayPrint(events);
        }
      else
        {
         PrintFormat("%s: CalendarEventByCountry(%s) 0 olay geri döndürdü, hata kodu=%
                     USA_code,__FUNCTION__,GetLastError());
         //--- operasyon bir hata ile tamamlandı, bir sonraki zamanlayıcı çağrısı sıra
         return;
        }

      //--- belirtilen olay için Takvim veritabanı change ID'sini elde et
      if(CalendarValueLastByEvent(event_id,calendar_change_id,values)>0)
        {
         //--- bu kod bloğu ilk çalıştırma sırasında yürütülemez ancak yine de ekleyel
         PrintFormat("%s: Takvim veritabanı mevcut ID'si elde edildi: change_id=%d",
                     __FUNCTION__,calendar_change_id);
         //--- bayrağı ayarla ve zamanlayıcının bir sonraki olayından önce çık
         first=false;
         return;
        }
      else
        {
         //--- veri alınmadı (bu ilk çalıştırma için normaldir), bir hata kontrolü yap
         int error_code=GetLastError();
         if(error_code==0)
           {
            PrintFormat("%s: Takvim veritabanı mevcut ID'si elde edildi: change_id=%d"
                        __FUNCTION__,calendar_change_id);
            //--- bayrağı ayarla ve zamanlayıcının bir sonraki olayından önce çık
            first=false;
            //--- şimdi calendar_change_id değerine sahibiz
            return;
           }
         else
           {
            //--- ve bu gerçekten bir hatadır
            PrintFormat("%s: event_id=%d için değerler alınamadı",__FUNCTION__,event_i
            PrintFormat("Hata kodu: %d",error_code);
            //--- operasyon bir hata ile tamamlandı, bir sonraki zamanlayıcı çağrısı s
            return;
           }
        }
     }

//--- Takvim change ID (change_id)'nin en son bilinen değerine sahibiz
```

```
   ulong old_change_id=calendar_change_id;
//--- yeni bir Nonfarm Payrolls olay değeri olup olmadığını kontrol et
   if(CalendarValueLastByEvent(event_id,calendar_change_id,values)>0)
     {
      PrintFormat("%s: \"%s\" için yeni olaylar alındı: %d",
                  __FUNCTION__,event_name,ArraySize(values));
      //--- Günlükteki 'değerler' dizisinden verileri görüntüle
      ArrayPrint(values);
      //--- Günlükteki önceki ve yeni Takvim ID değerlerini görüntüle
      PrintFormat("%s: Önceki change_id=%d, Yeni change_id=%d",
                  __FUNCTION__,old_change_id,calendar_change_id);
/*
      buraya, "Nonfarm Payrolls" olayının verilerini yönetmek için kodunuzu yazın
      */
     }
//---
  }
/*
  Sonuç:
   OnTimer: ABD olayları: 202
  "Nonfarm Payrolls" olayı bulundu: 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: Takvim veritabanı mevcut ID'si elde edildi: change_id=33986560

*/
```

Ayrıca bakınız

CalendarValueLast, CalendarValueHistory, CalendarValueHistoryByEvent, CalendarValueById

CalendarValueLast

Belirtilen bir change_id ile takvim veritabanı durumundan bu yana ülkeye ve/veya para birimine göre sıralanan tüm etkinlikler için değer dizisini elde edin.

```
int  CalendarValueLast(
   ulong&               change_id,              // change ID
   MqlCalendarValue&    values[],               // değer açıklamaları için dizi
   const string         country_code=NULL,      // kod olarak ülke adı (ISO 3166-1 alph
   const string         currency=NULL           // kod olarak ülke para birimi
   );
```

Parametreler

change_id

[in][out] Change ID.

values[]

[out] Olay değerlerini almak için MqlCalendarValue tip dizisi. Takvim olaylarını yönetme örneğine bakın.

country_code=NULL

[in] Kod olarak ülke adı (ISO 3166-1 alpha-2)

currency=NULL

[in] Kod olarak ülke para birimi.

Geri dönüş değeri

Alınan olay değeri sayısı. Hata hakkında bilgi edinmek için, GetLastError() fonksiyonunu çağırın. Olası hatalar:

- 4001 - ERR_INTERNAL_ERROR (genel çalışma zamanı hatası),
- 4004 - ERR_NOT_ENOUGH_MEMORY (istek yürütmek için yeterli bellek yok),
- 5401 - ERR_CALENDAR_TIMEOUT (istek süresi sınırı aşıldı),
- 5400 - ERR_CALENDAR_MORE_DATA (dizi boyutu, tüm değerlerin açıklamalarını almak için yetersiz; yalnızca diziye sığan açıklamalar alındı),
- ArrayResize()'ın başarısız yürütme hataları

Not

Ekonomik takvim ile çalışmak için tüm fonksiyonlar alım-satım sunucusu zamanını (TimeTradeServer) kullanır. Bu; MqlCalendarValue yapısındaki zamanın ve CalendarValueHistoryByEvent/CalendarValueHistory fonksiyonlarındaki zaman girdilerinin, kullanıcının yerel zamanından ziyade, alım-satım sunucusu zaman diliminde ayarlandığı anlamına gelir.

Eğer *events[]* sabit uzunluklu dizi fonksiyona iletildiyse ve sonucun tamamını kaydetmek için yeterli alan yoksa, ERR_CALENDAR_MORE_DATA (5400) hatası etkinleştirilir.

Eğer *change_id* = 0 fonksiyona iletilirse, fonksiyon her zaman sıfır geri döndürür, ancak geçerli takvim veritabanı *change_id* halinde geri döndürülür.

country_code ve *currency* filtreleri için NULL ve "" değerleri eşdeğerdir ve filtrenin yokluğu anlamına gelir.

country_code için, MqlCalendarCountry yapısının *kod* alanı, örneğin "US", "RU" veya "EU" kullanılmalıdır.

currency için, MqlCalendarCountry yapısının *currency* alanı, örneğin "USD", "RUB" veya "EUR" kullanılmalıdır.

Filtreler bir arada uygulanır, yani mantıksal 'VE' sadece her iki koşulun (ülke ve para birimi) aynı anda karşılandığı olayların değerlerini seçmek için kullanılır.

Fonksiyon, belirli bir haber için diziyi ve haberin yeni değerlerini almak adına fonksiyonun sonraki çağrıları için kullanılabilecek yeni bir *change_id* yi geri döndürür. Böylece, bu fonksiyonu bilinen en son *change_id* ile çağırarak belirli bir habere ait değerleri güncellemek mümkündür.

MqlCalendarValue yapısı, actual_value, forecast_value, prev_value ve revised_prev_value alanlarındaki değerleri kontrol etmek ve ayarlamak için yöntemler sağlar. Değer belirlenmezse, alan LONG_MIN (-9223372036854775808) olarak ayarlanır.

Lütfen bu alanlarda bulunan değerlerin bir milyon ile çarpıldığını unutmayın. Bunun anlamı, CalendarValueById, CalendarValueHistoryByEvent, CalendarValueHistory, CalendarValueLastByEvent ve CalendarValueLast fonksiyonlarını kullanarak MqlCalendarValue'da değerler aldığınızda, alandaki değerlerin LONG_MIN'e eşit olup olmadığını kontrol etmeniz; alanda bir değer belirlenmişse, değeri elde etmek için değeri 1.000.000'a bölmeniz gerektiği anlamına gelir. Değerleri elde etmenin diğer bir yöntemi de MqlCalendarValue yapısının fonksiyonlarını kullanarak değerleri kontrol etmek ve elde etmektir.

Ekonomik takvim olaylarını dinleyen örnek Uzman Danışman:

```
#property description "Ekonomik takvim olay dinleyicisini geliştirmek adına"
#property description " CalendarValueLast fonksiyonunu kullanım örneği"
#property description "Bunu başarmak için Takvim veritabanının"
#property description " mevcut change ID'sini elde edin. Devamında, zamanlayıcı anketi
#property description " almak için bu ID'yi kullanın"
//+------------------------------------------------------------------+
//| Uzman danışman başlatma fonksiyonu                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- zamanlayıcı oluştur
   EventSetTimer(60);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Uzman danışman sonlandırma fonksiyonu                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- zamanlayıcıyı kaldır
   EventKillTimer();
```

```
  }
//+------------------------------------------------------------------+
//| Uzman danışman tik fonksiyonu                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| Zamanlayıcı fonksiyonu                                           |
//+------------------------------------------------------------------+
void OnTimer()
  {
//--- Takvim veritabanı change ID'si
   static ulong calendar_change_id=0;
//--- ilk çalıştırmanın işareti
   static bool first=true;
//--- olay değeri dizisi
   MqlCalendarValue values[];
//--- başlatmayı gerçekleştir - mevcut calendar_change_id yi elde et
   if(first)
     {
      //--- Takvim veritabanı change ID'sini elde et
      if(CalendarValueLast(calendar_change_id,values)>0)
        {
         //--- bu kod bloğu ilk çalıştırma sırasında yürütülemez ancak yine de ekleyel
         PrintFormat("%s: Takvim veritabanı mevcut ID'si elde edildi: change_id=%d",
                     __FUNCTION__,calendar_change_id);
         //--- bayrağı ayarla ve zamanlayıcının bir sonraki olayından önce çık
         first=false;
         return;
        }
      else
        {
         //--- veri alınmadı (bu ilk çalıştırma için normaldir), bir hata kontrolü yap
         int error_code=GetLastError();
         if(error_code==0)
           {
            PrintFormat("%s: Takvim veritabanı mevcut ID'si elde edildi: change_id=%d'
                        __FUNCTION__,calendar_change_id);
            //--- bayrağı ayarla ve zamanlayıcının bir sonraki olayından önce çık
            first=false;
            //--- şimdi calendar_change_id değerine sahibiz
            return;
           }
         else
           {
            //--- ve bu gerçekten bir hatadır
```

```
            PrintFormat("%s: CalendarValueLast'ta olaylar alınamadı. Hata kodu: %d",
                        __FUNCTION__,error_code);
            //--- operasyon bir hata ile tamamlandı, bir sonraki zamanlayıcı çağrısı s
            return;
           }
        }
     }

//--- Takvim change ID (change_id)'nin en son bilinen değerine sahibiz
   ulong old_change_id=calendar_change_id;
//--- Yeni Takvim olayları olup olmadığını kontrol et
   if(CalendarValueLast(calendar_change_id,values)>0)
     {
      PrintFormat("%s: Yeni Takvim olayları alındı: %d",
                  __FUNCTION__,ArraySize(values));
      //--- Günlükteki 'değerler' dizisinden verileri görüntüle
      ArrayPrint(values);
      //--- Günlükteki önceki ve yeni Takvim ID değerlerini görüntüle
      PrintFormat("%s: Önceki change_id=%d, Yeni change_id=%d",
                  __FUNCTION__,old_change_id,calendar_change_id);
      //--- Günlükte yeni olayları görüntüle
      ArrayPrint(values);
      /*
     buraya, olayların meydana gelmesini yönetmek için kodunuzu yazın
      */
     }
//---
  }
/*
  Dinleyici operasyonuna örnek:
  OnTimer: Takvim veritabanı mevcut ID'si elde edildi: change_id=33281792
  OnTimer: Yeni Takvim olayları alındı: 1
        [id] [event_id]              [time]            [period] [revision] [actual_va
   [0] 91040   76020013 2019.03.20 15:30:00 1970.01.01 00:00:00          0       -507
  OnTimer: Önceki change_id=33281792, yeni change_id=33282048
        [id] [event_id]              [time]            [period] [revision] [actual_va
   [0] 91040   76020013 2019.03.20 15:30:00 1970.01.01 00:00:00          0       -507
  OnTimer: Yeni Takvim olayları alındı: 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: Önceki change_id=33282048, yeni 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

*/
```

Ayrıca bakınız

CalendarValueLast, CalendarValueHistory, CalendarValueHistoryByEvent, CalendarValueById

Zaman Serilerine ve Gösterge Verilerine Erişim

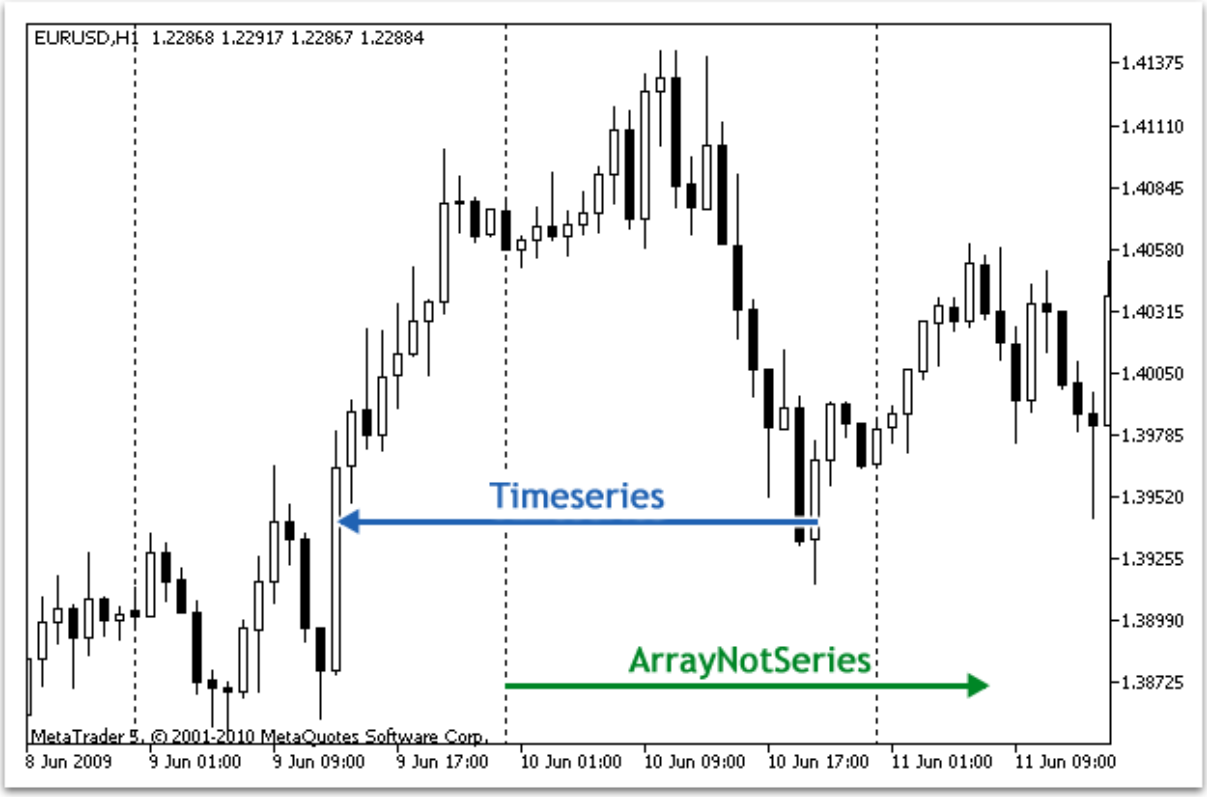
Bu fonksiyonlar, zaman-serileri ve göstergelerle çalışmak için tasarlanmıştır. Bu fonksiyonlar, zaman-serileri ve göstergelerle çalışmak için tasarlanmıştır. Zaman-serilerinin değerlerini ve gösterge verilerini kopyalamak için sadece dinamik dizilerin kullanılması önerilir, çünkü kopyalamayı yapan fonksiyonlar, değerlerin girileceği diziler için bellek tahsis etmek amacıyla tasarlanmışlardır.

Bu kuralın **önemli bir istisnası** bulunmaktadır: zaman-serilerinin ve gösterge verilerinin kopyalanmasına sıklıkla ihtiyaç duyuluyorsa (örneğin Uzman Danışman içindeki her OnTick() çağrısında veya gösterge içindeki her bir OnCalculate() çağrısı sırasında), bu durumda statik olarak dağıtılmış diziler kullanılmalıdır. Çünkü dinamik diziler için **bellek tahsisi işlemleri fazladan zaman gerektirir**, bu da sınama ve optimizasyon sırasında etkili olacaktır.

Zaman-serileri ve gösterge değerlerine erişmek amacıyla fonksiyonlar kullanıldığında, indisleme yönü de hesaba katılmalıdır. Bu konu, Dizilerde ve zaman-serilerinde indisleme yönü bölümünde açıklanmıştır.

Zaman-serileri ve gösterge verilerine yapılan erişim, verinin hazır olup olmamasına bakılmaksızın uygulanır (buna asenkron erişim) denir. Bu, özel göstergelerin hesaplanması için oldukça önemli bir özeliktir; eğer dizi içinde veri yoksa *Copy...()* tipindeki fonksiyonlar anında hata dönüşü yaparlar. Ama, scriptlerden ve Uzman Danışmanlardan erişim gerçekleştirirken, küçük bir duraklama içerisine verileri almak için birkaç deneme gerçekleştirilir. Bu duraklama, gereken zaman-serisi verilerinin alınması için yeterli sürenin sağlanması amacını taşımaktadır

Veri Erişiminin Düzenlenmesi bölümü, MetaTrader 5 müşteri terminali içerisinde fiyat verilerinin istenmesine, alınmasına ve saklanmasına dair detayları içermektedir.



Bir dizideki tarihsel verilere, verilerin sonundan başlayarak erişim gerçekleştirilir. Fiziksel olarak yeni veri, dizinin sonuna yazılır ve dizinin son elemanının indisi her zaman sıfıra eşittir. Zaman-serilerinde

0 indisi mevcut çubuğun verisini, yani mevcut zaman-aralığında tamamlanmamış olan çubuğa karşılık gelir.

Zaman-aralığı tek bir çubuğun şekillendiği zaman periyodudur. 21 adet ön-tanımlı standart zaman-aralığı bulunmaktadır.

Fonksiyon	Eylem
SeriesInfoInteger	Tarihsel verinin durumu ile ilgili veriye dönüş yapar
Bars	Belirtilen sembol ve periyot ile geçmişteki çubukların sayısına dönüş yapar
BarsCalculated	Bir gösterge tamponundaki hesaplanan verilerin sayısına dönüş yapar, hata durumunda ise -1 durumuna (veri henüz hesaplanmadı) dönüş yapar
IndicatorCreate	MqlParam tipi parametrelerden oluşmuş bir dizi ile oluşturulan belirli bir teknik göstergenin tanıtıcı değerine dönüş yapar
IndicatorParameters	Belirtilen tanıtıcı değere bağlı olarak, gösterge giriş parametrelerinin sayısına, değerlerine ve tiplerine dönüş yapar
IndicatorRelease	Gösterge tanıtıcı değerini kaldırır ve başka biri tarafından kullanılmıyorsa hesaplama bloğunu serbest bırakır
CopyBuffer	Belirtilen bir göstergenin belirtilen bir tamponunu bir dizi içine kopyalar
CopyRates	Belirtilen sembol ve periyot için Rates yapısının tarihsel verisini bir diziye kopyalar
CopySeries	Gets the synchronized timeseries from the Rates structure for the specified symbol-period and the specified amount
CopyTime	Belirtilen sembol ve periyot için çubuk açılış zamanına dair geçmiş verisini bir diziye kopyalar
CopyOpen	Belirtilen sembol ve periyot için çubuk açılış fiyatına dair geçmiş verisini bir diziye kopyalar
CopyHigh	Belirtilen sembol ve periyot için maksimal çubuk fiyatına dair geçmiş verisini bir diziye kopyalar
CopyLow	Belirtilen sembol ve periyot için minimal çubuk fiyatına dair geçmiş verisini bir diziye kopyalar
CopyClose	Belirtilen sembol ve periyot için çubuk kapanış fiyatına dair geçmiş verisini bir diziye kopyalar
CopyTickVolume	Belirtilen sembol ve periyot için tik hacmine dair geçmiş verisini bir diziye kopyalar
CopyRealVolume	Belirtilen sembol ve periyot için alım-satım hacmine dair geçmiş verisini bir diziye kopyalar
CopySpread	Belirtilen sembol ve periyot için makas değerine dair geçmiş verisini bir diziye kopyalar
CopyTicks	MqlTick biçimindeki tik verilerini ticks_array dizisine aktarır

Fonksiyon	Eylem
CopyTicksRange	belirtilen aralıktaki tik fiyatlarını MqlTick biçiminde ticks_array dizisine depolar
iBars	Tarihte mevcut karşılık gelen sembol ve dönemin çubuk sayısını döndürür
iBarShift	Zamana göre bar arama. İşlev, belirtilen zamana karşılık gelen çubuğun dizinini döndürür
iClose	İlgili grafikteki çubuğun Kapanış fiyatını ('shift' parametresiyle gösterilir) döndürür
iHigh	İlgili grafikteki çubuğun Yüksek fiyatını ('shift' parametresiyle gösterilir) döndürür
iHighest	İlgili grafikte bulunan en yüksek değerin indeksini döndürür (geçerli çubuğa göre)
iLow	İlgili grafikteki çubuğun Düşük fiyatını ('shift' parametresiyle gösterilir) döndürür
iLowest	İlgili grafikte bulunan en küçük değerin indeksini döndürür (geçerli çubuğa göre)
iOpen	İlgili grafikteki çubuğun Açılış fiyatını ('shift' parametresiyle gösterilir) döndürür
iTime	İlgili grafikteki çubuğun açılış zamanını ('shift' parametresiyle gösterilir) döndürür
iTickVolume	İlgili grafikteki çubuğun tik hacmini ('shift' parametresi ile gösterilir) döndürür
iRealVolume	İlgili grafikteki çubuğun gerçek hacmini ('shift' parametresiyle gösterilir) döndürür
iVolume	İlgili grafikteki çubuğun tik hacmini ('shift' parametresi ile gösterilir) döndürür
iSpread	İlgili grafikteki çubuğun spread değerini ('shift' parametresiyle gösterilir) döndürür

ArraySetAsSeries() fonksiyonunun kullanımıyla dizilerdeki veri erişiminin zaman-serilerindeki gibi ayarlanabilmesine rağmen, dizi elemanlarının aynı ve tek şekilde depolandıkları unutulmamalıdır - sadece indisleme yönü değişir. Bu özelliği gösterebilmek için şu örneği çalıştıralım:

```
   datetime TimeAsSeries[];
//--- diziye erişimi zaman-serilerindeki gibi ayarla
   ArraySetAsSeries(TimeAsSeries,true);
   ResetLastError();
   int copied=CopyTime(NULL,0,0,10,TimeAsSeries);
   if(copied<=0)
     {
      Print("Son 10 çubuğun açılış zamanını kopyalama işlemi başarısız oldu");
      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("Son 10 çubuğun açılış zamanını kopyalama işlemi başarısız oldu");
    return;
   }
 size=ArraySize(ArrayNotSeries);
 for(int i=size-1;i>=0;i--)
   {
    Print("ArrayNotSeries["+i+"] =",ArrayNotSeries[i]);
   }
```

Sonuç olarak şöyle bir çıktı alırız:

```
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
```

Çıktıdan da görebileceğimiz gibi, TimeAsSeries dizisinin indisi yükseldikçe, indisin zaman değeri azalıyor, Yani, şimdiki zamandan geçmişe doğru hareket ediyoruz. Normal ArrayNotSeries dizisi içinse sonuçlar farklı - indis büyürken, geçmişten günümüze hareket ediyoruz.

Ayrıca Bakınız

ArrayIsDynamic, ArrayGetAsSeries, ArraySetAsSeries, ArrayIsSeries

Dizilerde, Tamponlarda ve Zaman Serilerinde İndisleme Yönü

Tüm dizilerin varsayılan indisleme yönü soldan sağa şeklindedir. İlk elemanın indisi her zaman sıfıra eşittir. Bu nedenle, dizinin (veya tamponun) 0 indisli ilk elemanı varsayılan olarak en solda, son eleman ise en sağda yer alır.

Gösterge tamponu double tipli bir dinamik dizidir, büyüklüğü terminal tarafından ayarlanır ve her zaman göstergenin hesaplandığı çubuk sayısına denk gelir. Basit bir dinamik dizi, SetIndexBuffer() fonksiyonu kullanılarak gösterge tamponu olarak atanır. Gösterge tamponlarının boyutlarının ArrayResize() fonksiyonu ile yeniden ayarlanması gerekmez - bu işlem terminal idari alt-sistemi tarafından otomatik olarak gerçekleştirilir.

Zaman-serileri ters indisleme yönüne sahip dizilerdir, yani son eleman en solda, ilk eleman ise en sağda yer alır. Zaman-serileri, tarihsel fiyat verilerini ve zaman bilgisini depolamak için kullanılır. Bu şekilde baktığımızda en yeni verinin, zaman-serisinin en sağ konumunda yer aldığını, en eski verinin ise en sol konumda yer aldığını söyleyebiliriz.

Yani, 0 indisli zaman-serisi elemanı, sembolün son fiyatı hakkında bilgi içerir. Eğer bir zaman serisi günlük zaman aralığı verilerini içeriyorsa, mevcut (tamamlanmamış) günün verisi sıfır konumunda yer alır ve 1 indisine sahip konumda bir önceki günün verisi mevcuttur.

İndisleme yönünün değiştirilmesi

ArraySetAsSeries() fonksiyonu, dinamik dizi elemanlarının erişim yöntemini değiştirir; bu sırada bilgisayarın veriyi belleğe depolama yöntemi değişmez. Fonksiyon, dizi elemanlarının adresleme şeklini basitçe değiştirir, bu şekilde ArrayCopy() fonksiyonu ile başka bir diziye kopyalama yaparken, alıcı dizinin içeriği, kaynak diziye bağımlı olmaz.

İndisleme yönü statik olarak dağıtılmış dizilerde değiştirilemez. Dizi, fonksiyona parametre olarak geçirilmiş olsa bile, indisleme yönünün değiştirilmesinin hiç bir etkisi olmayacaktır.

Gösterge tamponlarının indisleme yönü de, tıpkı normal dizilerdeki gibi geriye doğru ayarlanabilir (zaman-serilerindeki gibi). Yani, göstergenin sıfır pozisyonuna yapılan referans, karşılık gelen tamponun son değerine, diğer bir deyişle, son çubuk üzerindeki gösterge değerine denk gelecektir. Buna rağmen, gösterge çubuklarının fiziksel yerleri değiştirilmeyecektir.

Göstergelerde Fiyat Verisinin Alınması

Her özel gösterge, hesaplamada kullanılacak fiyat verisinin geçirileceği OnCalculate() fonksiyonunu içermelidir. Geçirilen bu dizilerin indisleme yönü, ArrayGetAsSeries() fonksiyonu ile öğrenilebilir.

Fonksiyona geçirilen diziler, fiyat verilerini yansıtır; bu diziler zaman-serileri işaretine sahiptir ve ArrayIsSeries() fonksiyonu ile kontrol edildiklerinde, 'true' dönüşü alınır. Her durumda, indisleme yönünü öğrenmek için sadece ArrayGetAsSeries() fonksiyonu kullanılmalıdır.

Varsayılan değerlerden bağımsız olmak amacıyla, çalıştığınız diziler için ArraySetAsSeries() fonksiyonu koşulsuz olarak çağrılmalı ve istenen erişim yönü ayarlanmalıdır.

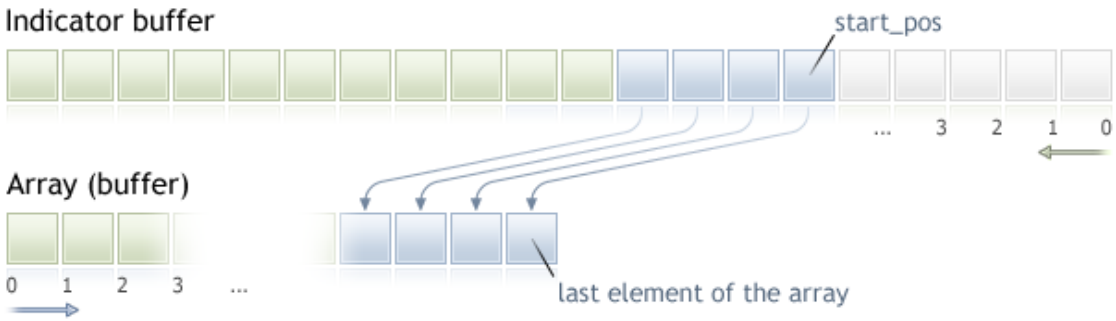
Fiyat Verisinin ve Gösterge Değerlerinin Alınması

Uzman Danışmanlarda, göstergelerde ve scriptlerde kullanılan ön-tanımlı indisleme yönü soldan-sağa doğrudur. Gerekli olduğu durumda, her MQL5 programı ve her sembol/periyot için zaman-serilerinin değerlerini isteyebilirsiniz.

Bu amaç için Copy...() fonksiyonunu kullanmalısınız:

- CopyBuffer - bir gösterge tamponunun verilerini double tipli bir diziye kopyalar;
- CopyRates - fiyat geçmişini MqlRates yapı tipli bir diziye kopyalar;
- CopyTime - Zaman değerlerini datetime tipi bir diziye kopyalar;
- CopyOpen - Open (açılış) fiyatlarını double tipli bir diziye kopyalar;
- CopyHigh - High (yüksek) fiyatlarını double tipli bir diziye kopyalar;
- CopyLow - Low (düşük) fiyatlarını double tipli bir diziye kopyalar;
- CopyClose - Close (kapanış) fiyatlarını double tipli bir diziye kopyalar;
- CopyTickVolume - tik hacmi değerlerini long tipli bir diziye kopyalar;
- CopyRealVolume - alım-satım hacmi değerlerini long tipli bir diziye kopyalar;
- CopySpread - makas geçmişini int tipli bir diziye kopyalar;

Tüm bu fonksiyonlar benzer şekilde çalışırlar. CopyBuffer() örneğindeki veri elde etme mekanizmasını düşünelim. İstenen verinin erişim şeklinin zaman serilerindeki gibi olduğu ve mevcut (tamamlanmamış) çubuğun verisinin 0 indisli konumda depolandığı gösterilmiştir. Bu verilere erişim için, gereken miktarda veriyi alıcı diziye kopyalamamız gerekir, örneğin *buffer* dizisi.



Kopyalama yapılırken kaynak dizideki başlangıç konumunu (verinin hangi konumdan başlanarak kopyalanacağını) belirtmemiz gerekir. Başarı durumunda, belirtilen sayıdaki eleman, kaynak diziden (gösterge tamponundan) alıcı diziye kopyalanacaktır. Alıcı dizide ayarlanan indisleme yönünden bağımsız olarak, kopyalama işlemi her zaman yukarıdaki resimde gösterildiği gibi gerçekleştirilir.

Fiyat verisinin yüksek sayıda tekrarlardan oluşan bir döngü ile işlenmesi bekleniyorsa, IsStopped() fonksiyonunu kullanarak programın zorla sonlandırılıp sonlandırılmadığını kontrol etmeniz önerilir:

```
int copied=CopyBuffer(ma_handle,// gösterge tanıtıcı değeri
                      0,        // Gösterge tamponunun indisi
                      0,        // Kopyalama için başlangıç pozisyonu
                      number,   // Kopyalanacak değerlerin sayısı
                      Buffer    // Kopyalanan değerlerin yazılacağı dizi
                      );
if(copied<0) return;
```

```
int k=0;
while(k<copied && !IsStopped())
  {
   //--- k indisi için değer al
   double value=Buffer[k];
   // ...
   // değerle çalış
   k++;
  }
```

Örnek:

```
input int per=10; // üs periyodu
int ma_handle;    // gösterge tanıtıcı değeri
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   ma_handle=iMA(_Symbol,0,per,0,MODE_EMA,PRICE_CLOSE);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---
   double ema[10];
   int copied=CopyBuffer(ma_handle,// gösterge tanıtıcı değeri
                         0,        // gösterge tamponunun indisi
                         0,        // kopyalama başlangıç konumu
                         10,       // kopyalanacak değerlerin sayısı
                         ema       // değerlerin yazılacağı dizi
                         );
   if(copied<0) return;
// .... daha sonraki kodlar
  }
```

Ayrıca Bakınız

Veri Erişiminin Düzenlenmesi

Veri Erişiminin Düzenlenmesi

Bu bölümde fiyat verilerinin (zaman-serileri) istenmesi, elde edilmesi ve depolanması ile ilgili sorular üzerinde durulacaktır.

Verinin Alım-Satım Sunucusundan Alınması

Fiyat verileri MetaTrader 5 terminalinde kullanılabilir duruma gelmeden önce alınıp işlenmelidir. Verilerin alınması için MetaTrader 5 alım-satım sunucusu ile bağlantı sağlanmalıdır. Terminalin isteğiyle birlikte, veriler dakikalık bloklar halinde paketlenmiş olarak alınır.

Veri için kullanılan sunucu referansının mekanizması, isteğin yapılma şeklinden (kullanıcı tarafından çizelgenin kaydırılmasıyla veya bir MQL5 programı kullanılarak) bağımsızdır.

Ara Verilerin Depolanması

Sunucudan alınan veriler otomatik olarak açılır ve HCC (ara) biçiminde saklanır. Her sembolün verileri ayrı klasörlere konulur: *terminal_dizini*\bases*sunucu_ismi*\history*sembol_ismi*. Örneğin, MetaQuotes-Demo sunucusundan alınan EURUSD verisi, *terminal_dizini*\bases\MetaQuotes-Demo\history\EURUSD\ konumunda saklanır.

Veriler, .hcc uzantılı dosyalara yazılırlar. Her bir dosya, bir senelik aralığı içerecek şekilde, dakikalık verilerden oluşur. Örneğin, EURUSD klasörü içinde 2009.hcc şeklinde isimlendirilmiş bir dosya, EURUSD sembolünün 2009 yılı için dakikalık verilerini içerecektir. Bu veriler tüm zaman-aralıkları için gereken fiyat verilerinin hazırlanması amacıyla kullanılırlar ve doğrudan kullanım için tasarlanmamışlardır.

Gerekli Zaman-aralığı Verilerinin Ara Verilerden Elde Edilmesi

HCC ara dosyaları, istenen zaman-aralığı verilerinin HC biçiminde kurulması amacıyla veri kaynağı şeklinde kullanılırlar. HC biçimli veriler, hızlı erişim için maksimal şekilde hazırlanmış zaman-serileridir. Bir çizelgenin veya bir MQL5 programının isteği üzerine oluşturulurlar. Verinin hacmi, çizelgede yer alabilecek maksimum çubuk sayısını ("Max bars in charts" parametresinin değerini) geçmemelidir. Veri, daha sonraki kullanımlar için hc uzantılı dosyalarda saklanır.

Kaynak tasarrufu amacıyla, zaman-aralığının verileri sadece gerektiğinde RAM üzerine kaydedilir. Uzun bir süre çağrılmazlarsa, RAM üzerinden silinip bir dosyaya kaydedilirler. Her bir zaman-aralığı için gereken veri, diğer zaman-aralıkları için de hazır olup olmadığına bakılmaksızın hazırlanır. Verilerin biçimlendirilmesine ve erişilmesine dair kurallar, her zaman-aralığı için aynıdır. Yani, HCC dosyasında saklanan veri biriminin "bir dakikalık" olması, HCC dosyasının varlığı, M1 zaman-aralığı verisinin HC biçiminde var olacağı anlamına gelmez

Sunucudan yeni verilerin alınmasıyla, tüm zaman-aralıklarındaki HC biçimli fiyat verileri otomatik olarak güncellenir. Bu durum, verileri arka-planda hesaplama için kullanan göstergelerin yeniden çizilmesine yol açar.

"Max bars in chart" Parametresi

"Max bars in charts" parametresi; çizelgeler, göstergeler ve MQL5 programları için kullanılabilir olan çubukların sayısını kısıtlar. Bu, tüm zaman aralıkları için geçerlidir ve her şeyden önce bilgisayar kaynaklarını korumak amacıyla tasarlanmıştır.

Bu parametre için büyük bir değer ayarlandığında, küçük zaman-aralıklarında kullanılan fiyat verileri için derin bir geçmiş mevcutsa, zaman-serilerinin ve gösterge tamponlarının depolanması için kullanılacak bellek miktarının yüzlerce megabayt olabileceği hatırlanmalıdır; bu durumda müşteri terminali için kullanılan RAM sınırlamalarına ulaşabilir (32-bit MS Windows uygulamaları için 2Gb).

"Max bars in charts" parametresinin değiştirilmesi, terminalin yeniden başlatılmasıyla etkinleşir. Bu parametrenin değiştirilmesi, ek veriler için otomatik olarak sunucuya başvurulmasına veya zaman serileri için ek çubukların şekillenmesine yol açmaz. Çizelgenin verilerin olmadığı bölüme kaydırılması veya bir MQL5 programının verileri istemesi durumunda, veriler sunucudan istenir ve zaman-serileri yeni kısıtlamalar doğrultusunda yeniden şekillenir.

Sunucudan istenecek veri miktarı, "Max bars in charts" parametresi göz önüne alınarak, zaman-aralığı için istenen çubuk sayısına karşılık gelecektir. Bu parametreyle ayarlanan sınırlama katı bir sınırlama değildir; bazı durumlarda bir zaman-aralığındaki mevcut çubukların sayısı, mevcut parametre değerini birazcık aşabilir.

Veri mevcudiyeti

Bir verinin HCC biçimindeki varlığı (HC biçiminde kullanılması için hazırlanmış olsa bile), bu verinin bir çizelge üzerinde gösterildiğini veya bir MQL5 programında kullanıldığını göstermez

Fiyat verilerine veya bir MQL5 programından gösterge değerlerine erişim gerçekleştirirken, verilerin belli bir andaki mevcudiyeti garanti edilmez. Kaynakların korunması amacıyla, MetaTrader 5 içerisinde MQL5 programı için gereken verinin tam bir kopyası depolanmaz; sadece terminalin veri tabanına doğrudan erişim sağlanır.

Tüm zaman-aralıklarındaki fiyat geçmişleri HCC ara biçimi temelinde kurulur ve sunucudan yapılacak her veri güncellemesi, tüm zaman-aralıklarının da güncellenmesine ve göstergelerin yeniden hesaplanmasına yol açar. Bunun bir sonucu olarak, veri erişimi - az önce mevcut olsa bile - bloke edilebilir.

Terminal Verilerinin Sunucu Verileri ile Senkronizasyonu

MQL5 programı herhangi bir sembolün/zaman-aralığının verisini çağırabileceği için, gerekli zaman-serilerinin, terminalde henüz hazırlanmamış olması veya gerekli fiyat verilerinin alım-satım sunucusu ile senkronize edilmemiş olması mümkündür. Bu durumda ortaya çıkacak gecikme anını tahmin edebilmek zordur.

Gecikme döngülerini kullanan algoritmalar en iyi çözümü vermezler. Bu durumun tek istisnası scriptlerdir; scriptler olay işleyicilerine sahip olmadıklarından alternatif bir algoritma seçenekleri yoktur. Özel göstergeler için benzer algoritmalar kesinlikle önerilmez, çünkü bu çeşit döngüler, mevcut göstergelerin hesaplama süreçlerinin ve sembole dair diğer fiyat-verisi işlemlerinin sonlandırılmasına yol açar.

Uzman Danışmanlar ve göstergeler için olay işleme modelinin kullanılması daha iyi olacaktır. OnTick() veya OnCalculate() olayının işlenmesi sırasında, gereken zaman-serisi için veri alınamazsa, işleyicinin bir sonraki çağrısı sırasındaki erişim durumuna dayanarak, olay işleyicisinden çıkılmalıdır.

Geçmiş Verilerinin Eklenmesi için bir Script Örneği

Alım-satım sunucusundan, seçilen sembolün geçmiş verisini almak için bir isteği uygulayan bir script örneği. Bu script, seçilen sembolün bir çizelgesi üzerinde çalıştırılmak için düşünülmüştür. Zaman-

aralığının bir önemi yoktur; yukarıda da bahsedildiği gibi, fiyat verileri sunucudan bir-dakikalık paketler şeklinde indirilir ve istenen zaman-aralıkları bu paket verilerden hesaplanır.

Veri alımına dair her eylemi ayrı bir fonksiyon şeklinde yazalım, CheckLoadHistory(symbol, timeframe, start_date):

```
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
  {
  }
```

CheckLoadHistory() fonksiyonu, herhangi bir programdan (Uzman Danışman, script veya gösterge) çağrılabilecek evrensel bir fonksiyon olarak tasarlanmıştır; üç giriş parametresine ihtiyaç duyar: sembol ismi, periyot ve (istenilen fiyat geçmişinin başlangıcını belirtmek için) başlangıç tarihi.

Eksik veriyi istemeden önce fonksiyon kodundaki gerekli yerleri dolduralım. Öncelikle sembol isminin ve periyot değerinin doğruluğundan emin olalım:

```
   if(symbol==NULL || symbol=="") symbol=Symbol();
   if(period==PERIOD_CURRENT)     period=Period();
```

Ardından, sembolün Piyasa gözlemi Penceresinde yer aldığını doğrulayalım, yani sunucuya istek gönderdiğimizde sembolün geçmişi mevcut olsun. Piyasa Gözleminde böyle bir sembol bulunmuyorsa, SymbolSelect() fonksiyonunu kullanarak ekleyelim.

```
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
     {
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
     }
```

Şimdi, istenen sembol/periyot çifti için mevcut geçmişin başlangıç tarihini almalıyız. CheckLoadHistory() fonksiyonuna geçirilen startdate giriş parametresinin değeri, mevcut geçmiş içerisinde yer alıyor olabilir; bu durumda sunucudan veri isteği gereksiz olacaktır. Seçilen sembol/periyot için, mevcut olan ilk tarihi öğrenmek amacıyla, SeriesInfoInteger() fonksiyonu, SERIES_FIRSTDATE şekillendiricisi ile birlikte kullanılır.

```
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
```

Bir sonraki önemli adım, fonksiyonun çağrılacağı programın tipinin kontrol edilmesidir. Aynı periyoda sahip gösterge zaman-serilerinin güncellenmesi için yapılan isteklerin sakıncalı olacağını not ediniz. Bu tip güncellemeler, göstergenin çalışmakta olduğu iş parçacığı üzerinde gerçekleştirilir. Bu nedenle, bir çakışma gerçekleşmesi yüksek olasılıktır. Bunu kontrol etmek için MQL5InfoInteger() fonksiyonunu MQL5_PROGRAM_TYPE şekillendiricisi ile kullanın.

```
   if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Sy
      return(-4);
```

Eğer tüm kontroller başarıyla gerçekleştirilmişse, alım-satım sunucusuna başvurmadan son bir deneme yapalım. İlk önce, HCC biçimli mevcut dakikalık veriler için başlangıç tarihini öğrenelim. Bu değeri SeriesInfoInteger() fonksiyonunu SERIES_TERMINAL_FIRSTDATE şekillendiricisi ile kullanarak alırız ve yine start_date parametresinin değeriyle karşılaştırırız.

```
   if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
```

```
     {
      //--- zaman-serilerinin kurulabileceği veri mevcut
      if(first_date>0)
        {
         //--- zaman-serilerini kurmaya zorla
         CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
         //--- tarihi kontrol et
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(2);
        }
     }
```

Tüm kontrollerden sonra uygulama iş parçacığı hala CheckLoadHistory() fonksiyonunun gövdesinde yer alıyorsa, eksik veri, alım-satım sunucusundan istenmelidir. Bunun için önce, "Max bars in chart" değerine TerminalInfoInteger() fonksiyonu ile dönüş yapalım:

```
   int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
```

Bu veri fazladan veri istememiz için gereklidir. Ardından, sembolün geçmişinin alım-satım sunucusunda bulunan ilk tarihini (regardless of the period), SERIES_SERVER_FIRSTDATE şekillendiricisi ile SeriesInfoInteger() fonksiyonunu kullanarak öğrenmeliyiz.

```
    datetime first_server_date=0;
    while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date
       Sleep(5);
```

İsteğin asenkron bir işlem olması nedeniyle, fonksiyon, bir döngü içerisinde (first_server_date değişkeni bir değer alana kadar) beş milisaniyelik gecikme ile çağrılır veya döngü işlemi kullanıcı tarafından sonlandırılır; bu durumda IsStopped() fonksiyonu, *true* dönüşü yapacaktır. Alım-satım sunucusundan alacağımız veriler için geçerli bir değer belirleyelim.

```
    if(first_server_date>start_date) start_date=first_server_date;
    if(first_date>0 && first_date<first_server_date)
       Print("Uyarı: ilk sunucu tarihi ",first_server_date,,
symbol," için serinin ilk tarihiyle uyumlu değil ",first_date);
```

Sunucudaki başlangıç tarihinin (*first_server_date*), sembolün HCC biçimindeki başlangıç tarihinden (*first_date*) düşük olması durumunda, karşılık gelen giriş bültene çıktılanır.

Artık, eksik fiyat verisini alım-satım sunucusundan istemek için hazırız. İsteği bir döngü şeklinde gerçekleştir, söz konusu döngünün gövdesini doldurmaya başla:

```
    while(!IsStopped())
      {
       //1. yeniden kurulan zaman-serisi ile HHC biçimindeki ara verinin senkronizasyo
       //2. söz konusu zaman-serisindeki mevcut çubuk sayısını al
       //    çubuk sayısı, "Max_bars_in_chart" paraetresinden büyükse, çıkabiliriz, iş
       //3. yeniden kurulan zaman-serisinin first_date başlangıç tarihini al ve start_
       //    first_date değeri, start_date değerinden küçükse çıkabiliriz, iş bitti
       //4. sunucudan, geçmişin yeni bir parçasını iste - 'bars' numaralı son çubuktan
      }
```

İlk üçü, zaten bilinen araçlarla uygulanır.

```
   while(!IsStopped())
     {
      //--- 1.zaman-serisi yeniden kurulanan kadar bekle
      while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
         Sleep(5);
      //--- 2.kaç çubuğumuz var öğren
      int bars=Bars(symbol,period);
      if(bars>0)
        {
         //--- çubuklar, çizelgeye çizilebilecek miktardan fazla, çık
         if(bars>=max_bars) return(-2);
         //--- 3. zaman-serisindeki mevcut başlangıç tarihine dönüş yap
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            // başlangıç tarihi, istenenden daha önceye denk geliyor, görev tamamlandı
            if(first_date>0 && first_date<=start_date) return(0);
        }
      //4. Geçmişin yeni bir parçasını iste - 'bars' numaralı son çubuktan başlayarak
     }
```

son dördüncü nokta kaldı (geçmiş isteği). Sunucuya doğrudan başvuramayız ama HCC içindeki geçmiş yetersiz ise herhangi bir Copy-fonksiyonu ile sunucuya istek gönderme işlemini otomatik olarak başlatabiliriz. *first_date* değişkeni içindeki başlangıç tarihi, istek uygulama derecesi için basit ve doğal bir kriter olduğundan, kullanılabilecek en kolay yol CopyTime() fonksiyonudur.

Zaman-serilerinden veri kopyalayan fonksiyonlar çağrıldığında, *start* parametresinin (kopyalamanın başlayacağı çubuk numarası) hiç bir zaman mevcut terminal geçmişinin dışında olmaması gerektiği not edilmelidir. Sadece 100 çubuğa sahipseniz, 500 numaralı çubuktan başlayarak 300 çubuk kopyalamayı denenmenin hiç bir anlamı olmaz. Böyle bir istek, hatalı olarak değerlendirilecek ve işlenmeyecektir, yani alım-satım sunucusundan fazladan veri yüklenmeyecektir.

Bu yüzden *bars* indisli çubuktan başlayarak 100 çubuk kopyalarız. Bu, eksik verilerin alım-satım sunucusundan düzgünce yüklenmesini sağlayacaktır. Aslında, sunucu fazladan veriler gönderir, böylece 100 çubuktan fazla yüklenecektir.

```
   int copied=CopyTime(symbol,period,bars,100,times);
```

Kopyalama işleminden sonra, kopyalanan elemanların sayısını kontrol etmeliyiz. Girişim başarısız olursa, *copied* değeri null olacak ve fail_cnt sayacının değeri bir artırılacaktır. 100 başarısız denemeden sonra, fonksiyonun işlemi sonlandırılacaktır.

```
int fail_cnt=0;
...
   int copied=CopyTime(symbol,period,bars,100,times);
   if(copied>0)
     {
      //--- veriyi kontrol et
      if(times[0]<=start_date)  return(0);  // kopyalanan veri daha küçük, hazır
      if(bars+copied>=max_bars) return(-2); // çubuklar çizelgeye çizilebilecek olanda
      fail_cnt=0;
```

```
    }
  else
    {
     //--- 100'den az denemede başarı
     fail_cnt++;
     if(fail_cnt>=100) return(-5);
     Sleep(10);
    }
```

Böylece, hem çalışmanın her anındaki mevcut durum doğru şekilde işlenir, hem de CheckLoadHistory() fonksiyonunun çağrısından sonra, ek bilgiler için işlenebilecek olan sonlandırma koduna dönüş yapılır. Örneğin, bu yol:

```
  int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
  switch(res)
    {
     case -1 : Print("Bilinmeyen sembol ",InpLoadedSymbol);                  break;
     case -2 : Print("Çizelgeye çizilebilecek olandan daha fazla istenen çubuk var");
     case -3 : Print("Uygulama kullanıcı tarafından durduruldu");           break;
     case -4 : Print("Gösterge, kendi verisini yüklememeli");               break;
     case -5 : Print("Yükleme başarısız");                                  break;
     case  0 : Print("Tüm veriler yüklendi");                               break;
     case  1 : Print("Zaman-serisindeki zaten yüklü olan veriler yeterli"); break;
     case  2 : Print("Zaman-serisi, terminaldeki mevcut verilerden kuruldu"); break;
     default : Print("Uygulama sonucu başarısız");
    }
```

Fonksiyonun bütün kodu, herhangi bir veriye yapılan erişimin doğru şekilde düzenlenmesini ve istek sonuçlarının işlenmesini gösteren bir script örneğinde bulunabilir.

Code:

```
//+------------------------------------------------------------------+
//|                                              TestLoadHistory.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.02"
#property script_show_inputs
//--- giriş parametreleri
input string          InpLoadedSymbol="NZDUSD";   // Yüklenecek sembol
input ENUM_TIMEFRAMES InpLoadedPeriod=PERIOD_H1;  // Yüklenecek periyot
input datetime        InpStartDate=D'2006.01.01'; // Başlangıç tarihi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

```
   Print("Yüklemeyi başlat",InpLoadedSymbol+","+GetPeriodName(InpLoadedPeriod),"tarihi
//---
   int res=CheckLoadHistory(InpLoadedSymbol,InpLoadedPeriod,InpStartDate);
   switch(res)
     {
      case -1 : Print("Bilinmeyen sembol ",InpLoadedSymbol);              break;
      case -2 : Print("Çizelgeye çizilebilecek olandan daha fazla istenen çubuk var");
      case -3 : Print("Program durduruldu");                            break;
      case -4 : Print("Gösterge, kendi verisini yüklememeli");          break;
      case -5 : Print("Yükleme başarısız");                                break;
      case  0 : Print("Yükleme başarılı");                                 break;
      case  1 : Print("Önceden yüklenmiş");                             break;
      case  2 : Print("Önceden yüklenmiş ve kurulmuş");                   break;
      default : Print("Bilinmeyen sonuç");
     }
//---
   datetime first_date;
   SeriesInfoInteger(InpLoadedSymbol,InpLoadedPeriod,SERIES_FIRSTDATE,first_date);
   int bars=Bars(InpLoadedSymbol,InpLoadedPeriod);
   Print("İlk tarih ",first_date," - ",bars," çubuk");
//---
  }
//+------------------------------------------------------------------+
//|                                                                  |
//+------------------------------------------------------------------+
int CheckLoadHistory(string symbol,ENUM_TIMEFRAMES period,datetime start_date)
  {
   datetime first_date=0;
   datetime times[100];
//--- sembol ve periyodu kontrol et
   if(symbol==NULL || symbol=="") symbol=Symbol();
   if(period==PERIOD_CURRENT)     period=Period();
//--- sembol Piyasa Gözleminde seçili mi kontrol et
   if(!SymbolInfoInteger(symbol,SYMBOL_SELECT))
     {
      if(GetLastError()==ERR_MARKET_UNKNOWN_SYMBOL) return(-1);
      SymbolSelect(symbol,true);
     }
//--- veri mevcut mu kontrol et
   SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date);
   if(first_date>0 && first_date<=start_date) return(1);
//--- bu bir gösterge ise kendi verisini yüklemesini isteme
   if(MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR && Period()==period && Sy
      return(-4);
//--- ikinci deneme
   if(SeriesInfoInteger(symbol,PERIOD_M1,SERIES_TERMINAL_FIRSTDATE,first_date))
     {
      //--- zaman-serilerinin kurulabileceği veri mevcut
      if(first_date>0)
```

```
         {
          //--- zaman-serilerini kurmaya zorla
          CopyTime(symbol,period,first_date+PeriodSeconds(period),1,times);
          //--- tarihi kontrol et
          if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
             if(first_date>0 && first_date<=start_date) return(2);
         }
      }
//--- terminal seçeneklerinden "max bars in chart"
   int max_bars=TerminalInfoInteger(TERMINAL_MAXBARS);
//--- sembol geçmişinin bilgisini yükle
   datetime first_server_date=0;
   while(!SeriesInfoInteger(symbol,PERIOD_M1,SERIES_SERVER_FIRSTDATE,first_server_date
      Sleep(5);
//--- yükleme için başlangıç tarihini düzelt
   if(first_server_date>start_date) start_date=first_server_date;
   if(first_date>0 && first_date<first_server_date)
      Print("Uyarı: ilk sunucu tarihi ",first_server_date,,symbol,
            " için serinin ilk tarihiyle uyumlu değil ",first_date);
//--- veriyi adım adım yükle
   int fail_cnt=0;
   while(!IsStopped())
     {
      //--- zaman-serisinin kurulması için bekle
      while(!SeriesInfoInteger(symbol,period,SERIES_SYNCHRONIZED) && !IsStopped())
         Sleep(5);
      //--- kurulan çubukları iste
      int bars=Bars(symbol,period);
      if(bars>0)
        {
         if(bars>=max_bars) return(-2);
         //--- ilk tarihi iste
         if(SeriesInfoInteger(symbol,period,SERIES_FIRSTDATE,first_date))
            if(first_date>0 && first_date<=start_date) return(0);
        }
      //--- bir sonraki kısmın kopyalanması, verinin yüklenmesi için zorlar
      int copied=CopyTime(symbol,period,bars,100,times);
      if(copied>0)
        {
         //--- veriyi kontrol et
         if(times[0]<=start_date)  return(0);
         if(bars+copied>=max_bars) return(-2);
         fail_cnt=0;
        }
      else
        {
         //--- 100'den az denemede başarı
         fail_cnt++;
         if(fail_cnt>=100) return(-5);
```

```
         Sleep(10);
        }
      }
//--- durduruldu
   return(-3);
  }
//+------------------------------------------------------------------+
//| Periyodun string tipli değerine dönüş yapar                      |
//+------------------------------------------------------------------+
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("bilinmeyen periyot");
  }
```

SeriesInfoInteger

Tarihsel verinin durumu ile ilgili veriye dönüş yapar. Fonksiyon çağrıları için 2 versiyon bulunmaktadır.

Doğrudan özellik değerine dönüş yapar.

```
long  SeriesInfoInteger(
   string                      symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES             timeframe,        // periyot
   ENUM_SERIES_INFO_INTEGER    prop_id,          // özellik tanımlayıcısı
   );
```

Fonksiyonun başarı durumuna göre, 'true' veya 'false' değerine dönüş yapar.

```
bool  SeriesInfoInteger(
   string                      symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES             timeframe,        // periyot
   ENUM_SERIES_INFO_INTEGER    prop_id,          // özellik tanımlayıcısı
   long&                       long_var          // özelliğin alınması için hedef değişk
   );
```

Parametreler

symbol_name

[in] Sembol ismi.

timeframe

[in] Periyot.

prop_id

[in] İstenen özelliğin tanımlayıcısı, ENUM_SERIES_INFO_INTEGER sayımının değerlerinden biri.

long_var

[out] İstenen özellik değerinin yerleştirileceği değişken.

Dönüş değeri

İlk versiyonda long tipli bir değere dönüş yapar.

İkinci durumda, eğer belirtilen özellik mevcutsa ve bunun değeri long_var değişkenine girilmişse 'true' dönüşü, aksi durumda 'false' dönüşü yapar. Hata ile ilgili daha detaylı bilgi almak için GetLastError() fonksiyonunu çağırın.

Örnek:

```
void OnStart()
  {
//---
   Print("Mevcut sembol ve periyot için mevcut çubukların sayısı = ",
         SeriesInfoInteger(Symbol(),Period(),SERIES_BARS_COUNT));

   Print("Mevcut sembol ve periyot için geçmişteki ilk tarih = ",
         (datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_FIRSTDATE));
```

```
   Print("Mevcut sembol ve periyot için sunucudaki ilk tarih = ",
         (datetime)SeriesInfoInteger(Symbol(),Period(),SERIES_SERVER_FIRSTDATE));

   Print("Sembol verisi senkronize edilmiş = ",
         (bool)SeriesInfoInteger(Symbol(),Period(),SERIES_SYNCHRONIZED));
  }
```

Bars

Belli bir sembol ve periyot değri için geçmiş verilerde yer alan çubuk sayısını verir. Fonksiyon çağrısının iki versiyonu vardır.

Tüm geçmiş çubukların istenmesi

```
int  Bars(
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe         // periyot
   );
```

Belirli bir zaman aralığındaki çubukların istenmesi

```
int  Bars(
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   datetime         start_time,       // başlangıç tarihi ve zamanı
   datetime         stop_time         // son tarih ve zaman
   );
```

Parametreler

symbol_name

[in] Sembol ismi.

timeframe

[in] Periyot.

start_time

[in] İlk elemana karşılık gelen çubuğun zamanı.

stop_time

[in] Son elemana karşılık gelen çubuğun zamanı.

Dönüş Değeri

start_time ve stop_time parametreleri girilmişse belirtilen zaman aralığındaki çubuk sayısına, aksi durumda toplam çubuk sayısına dönüş yapar.

Not

Bars() fonksiyonunun çağrısı sırasında ilgili zaman serisinin verileri terminalde mevcut değilse veya sunucuyla eşitlenmemişse fonksiyon '0' dönüşü yapar.

Belli bir zaman aralığındaki çubukların sayısı istendiğinde, sadece açılış zamanları bu aralığa düşen çubuklar hesaba katılır. Örneğin, Cumartesi günü start_time=son_salı ve stop_time=son_cuma parametreleriyle W1 çubuklarının sayısı istenirse fonksiyon '0' dönüşü yapacaktır. Çünkü W1 periyodu için açılış zamanı Pazar günüdür, dolayısıyla ilgili zaman aralığında herhangi bir W1 çubuğu yoktur.

Tüm geçmiş veriler için istek örneği:

```
   int bars=Bars(_Symbol,_Period);
   if(bars>0)
```

```
   {
    Print("Bu sembol ve periyot için geçmiş veride yer alan çubuk sayısı = ",bars);
   }
 else  //hiç çubuk yok
   {
    //--- sembol verisi sunucu verileriyle eşitlenmemiş olabilir
    bool synchronized=false;
    //--- döngü sayacı
    int attempts=0;
    // eşitleme için 5 deneme yap
    while(attempts<5)
      {
       if(SeriesInfoInteger(Symbol(),0,SERIES_SYNCHRONIZED))
         {
          //--- eşitleme bitti, çık
          synchronized=true;
          break;
         }
       //--- sayacı artır
       attempts++;
       //--- sonraki tekrara kadar 10 milisaniye bekle
       Sleep(10);
      }
    //--- eşitlemenin ardından döngüyü bitir
    if(synchronized)
      {
       Print("Bu sembol ve periyot için terminal geçmişindeki çubuk sayısı = ",bars)
       Print("Bu sembol ve periyot için terminal geçmişindeki ilk tarih = ",
             (datetime)SeriesInfoInteger(Symbol(),0,SERIES_FIRSTDATE));
       Print("Bu sembol için sunucuda bulunan ilk tarih = ",
             (datetime)SeriesInfoInteger(Symbol(),0,SERIES_SERVER_FIRSTDATE));
      }
    //--- veriler eşitlenemedi
    else
      {
       Print(_Sembol," için çubuk sayısı alınamadı");
      }
   }
```

Belirtilen aralık için çubuk sayısının istenmesi örneği:

```
int n;
datetime date1 = D'2016.09.02 23:55'; // Cuma
datetime date2 = D'2016.09.05 00:00'; // Pazartesi
datetime date3 = D'2016.09.08 00:00'; // Perşembe
//---
n=Bars(_Symbol,PERIOD_H1,D'2016.09.02 02:05',D'2016.09.02 10:55');
Print("Çubuk sayısı: ",n); // Çıktı: "Çubuk sayısı: 8", H2 hesaba dahil edildi, H1
n=Bars(_Symbol,PERIOD_D1,date1,date2);
```

```
   Print("Çubuk sayısı: ",n); // Çıktı: "Çubuk sayısı: 1", belirtilen aralıkta bir D1
   n=Bars(_Symbol,PERIOD_W1,date2,date3);
   Print("Çubuk sayısı: ",n); // Çıktı: "Çubuk sayısı: 0", belirtilen aralıkta herhan
```

Ayrıca bakınız

Olay İşleyici Fonksiyonları

BarsCalculated

Belirtilen gösterge için hesaplanan verilerin sayısına dönüş yapar.

```
int  BarsCalculated(
   int       indicator_handle,     // gösterge tanıtıcı değeri
   );
```

Parametreler

indicator_handle

[in] Karşılık gelen fonksiyonla dönüşü yapılan gösterge tanıtıcı değeri.

Dönüş değeri

Belirtilen gösterge için hesaplanan verilerin sayısına, hata durumunda ise -1 değerine (veri henüz hesaplanmamış) değerine dönüş yapar

Not

Bu fonksiyon, gösterge verisinin göstergenin oluşturulmasının hemen ardından alınması için kullanışlıdır.

Örnek:

```
void OnStart()
  {
   double Ups[];
//--- dizilerin indis yönünü zaman-serilerindeki gibi ayarla
   ArraySetAsSeries(Ups,true);
//--- Fractal göstergesi için bir tanıtıcı değer oluştur
   int FractalsHandle=iFractals(NULL,0);
//--- hata kodunu sıfırla
   ResetLastError();
//--- gösterge değerlerini kopyalamayı dene
   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("Üst fraktallerin kopyalanması başarısız. Hata = ",GetLastError(),
         "i = ",i,"    copied = ",copied);
         return;
        }
```

```
      else
        Print("Üst fraktaller kopyalandı",
        "i = ",i,"    copied = ",copied);
    }
  else Print("Üst fraktaller kopyalandı. ArraySize = ",ArraySize(Ups));
}
```

IndicatorCreate

MqlParam tipi parametrelerden meydana gelen bir diziyle oluşturulmuş teknik göstergenin tanıtıcı değerine dönüş yapar.

```
int  IndicatorCreate(
   string           symbol,                    // sembol ismi
   ENUM_TIMEFRAMES  period,                    // zaman-aralığı
   ENUM_INDICATOR   indicator_type,            // ENUM_INDICATOR içinden seçil
   int              parameters_cnt=0,          // parametrelerin sayısı
   const MqlParam&  parameters_array[]=NULL,   // parametrelerin dizisi
   );
```

Parametreler

symbol

[in] Gösterge verisinin hesaplandığı sembolün ismi. NULL mevcut sembol anlamına gelir.

period

[in] Zaman-aralığı değeri, ENUM_TIMEFRAMES sayımının değerlerinden biri olabilir, 0 mevcut zaman-aralığı anlamına gelir.

indicator_type

[in] Gösterge tipi, ENUM_INDICATOR sayımının değerlerinden biri olabilir.

parameters_cnt

[in] parameters_array[] dizisine geçirilen parametrelerin sayısı. Dizi elemanları özel bir yapı tipine sahiptir: MqlParam. Varsayılan değeri sıfırdır (parametreler geçirilmedi). Sıfır harici bir parametre sayısını belirtmeniz durumunda, *parameters_array* kullanımı zorunludur. Diziye 64'dan fazla parametre geçiremezsiniz.

parameters_array[]=NULL

[in] MqlParam tipi bir dizi - elemanları, bir teknik göstergenin her bir giriş parametresinin tipini ve değerini içerir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar.

Not

IND_CUSTOM tipinde bir gösterge oluşturulmuşsa, *parameters_array* giriş parametrelerinden oluşan dizinin ilk elemanının *type* alanı, ENUM_DATATYPE sayımının TYPE_STRING değerini, *string_value* alanı ise özel göstergenin ismini içermelidir. Özel gösterge (EX5 uzantılı dosya) derlenmeli ve MQL5/Indicators konumunda veya bir alt klasörde yer almalıdır.

Sınama gerektiren göstergeler, eğer karşılık gelen parametre bir sabit dizgi ile ayarlanmışsa, iCustom() fonksiyonunun çağrısıyla otomatik olarak tanımlanırlar. Tüm diğer durumlarda (IndicatorCreate() fonksiyonunun kullanımı veya sabit olmayan bir dizginin gösterge ismini belirleyen bir parametrede kullanımı) #property tester_indicator özelliği gereklidir:

```
#property tester_indicator "gösterge_ismi.ex5"
```

Bir özel gösterge içerisinde ilk çağrı versiyonunun kullanılması durumunda, giriş parametrelerini geçirirken son parametre ayrı olarak belirtilebilir. "Apply to" parametresi açık şekilde belirtilmemişse, varsayılan hesaplama PRICE_CLOSE değerlerinin temelinde gerçekleştirilir.

Örnek:

```
void OnStart()
  {
   MqlParam params[];
   int      h_MA,h_MACD;
//--- iMA("EURUSD",PERIOD_M15,8,0,MODE_EMA,PRICE_CLOSE) göstergesini oluştur;
   ArrayResize(params,4);
//--- ma_period değerini ayarla
   params[0].type         =TYPE_INT;
   params[0].integer_value=8;
//--- ma_shift değerini ayarla
   params[1].type         =TYPE_INT;
   params[1].integer_value=0;
//--- ma_method değerini ayarla
   params[2].type         =TYPE_INT;
   params[2].integer_value=MODE_EMA;
//--- applied_price değerini ayarla
   params[3].type         =TYPE_INT;
   params[3].integer_value=PRICE_CLOSE;
//--- MA göstergesini oluştur
   h_MA=IndicatorCreate("EURUSD",PERIOD_M15,IND_MA,4,params);
//--- iMACD("EURUSD",PERIOD_M15,12,26,9,h_MA) göstergesini oluştur;
   ArrayResize(params,4);
//--- hızlı ma_period değerini ayarla
   params[0].type         =TYPE_INT;
   params[0].integer_value=12;
//--- yavaş ma_period değerini ayarla
   params[1].type         =TYPE_INT;
   params[1].integer_value=26;
//--- fark için düzleştirme periyodunu ayarla
   params[2].type         =TYPE_INT;
   params[2].integer_value=9;
//--- gösterge tanıtıcı değerini applied_price şeklinde ayarla
   params[3].type         =TYPE_INT;
   params[3].integer_value=h_MA;
//--- MACD göstergesini hareketli ortalama temelinde oluştur
   h_MACD=IndicatorCreate("EURUSD",PERIOD_M15,IND_MACD,4,params);
//--- göstergeleri kullan
//--- . . .
//--- göstergeleri serbest bırak (önce h_MACD)
   IndicatorRelease(h_MACD);
   IndicatorRelease(h_MA);
  }
```

IndicatorParameters

Belirtilen tanıtıcı değere bağlı olarak, gösterge giriş parametrelerinin sayısına, değerlerine ve tiplerine dönüş yapar.

```
int  IndicatorParameters(
   int               indicator_handle,     // gösterge tanıtıcı değeri
   ENUM_INDICATOR&   indicator_type,       // gösterge tipini almak için bir değişken
   MqlParam&         parameters[]          // parametreleri almak için bir dizi
   );
```

Parametreler

indicator_handle

[in] Parametre sayısının öğrenilmek istendiği göstergenin tanıtıcı değeri.

indicator_type

[out] Gösterge tipinin depolanması için ENUM_INDICATOR tipi bir değişken.

parameters[]

[out] Gösterge parametrelerinin listesinin kaydedileceği, MqlParam tipi değerlerden oluşan bir dinamik dizi. Dizi büyüklüğü IndicatorParameters() fonksiyonu ile alınabilir.

Dönüş değeri

Belirtilen tanıtıcı değere sahip göstergenin giriş parametrelerinin sayısı. Hata durumunda -1 dönüşü yapar. Hata hakkında daha çok bilgi için GetLastError() fonksiyonunu çağırın.

Örnek:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {

//--- Çizelgedeki pencerelerin sayısı (en azından bir ana pencere her zaman mevcuttur)
   int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
//--- Çizelge pencerelerini incele
   for(int w=0;w<windows;w++)
     {
      //--- Söz konusu (alt)penceredeki göstergelerin sayısı
      int total=ChartIndicatorsTotal(0,w);
      //--- Penceredeki tüm göstergeleri alır
      for(int i=0;i<total;i++)
        {
         //--- Göstergenin kısa ismini al
         string name=ChartIndicatorName(0,w,i);
         //--- Gösterge tanıtıcı değerini al
         int handle=ChartIndicatorGet(0,w,name);
         //--- Günlüğe ekle
         PrintFormat("Pencere=%d,  gösterge #%d,  tanıtıcı değer=%d",w,i,handle);
```

```
         //---
         MqlParam parameters[];
         ENUM_INDICATOR indicator_type;
         int params=IndicatorParameters(handle,indicator_type,parameters);
         //--- Mesajın tanıtıcı değeri
         string par_info="Kısa isim "+name+", tip "
                         +EnumToString(ENUM_INDICATOR(indicator_type))+"\r\n";
         //---
         for(int p=0;p<params;p++)
           {
            par_info+=StringFormat("parametre %d: tip=%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);
        }
      //--- Penceredeki tüm göstergelere uygulandı
     }
//---
  }
```

Ayrıca Bakınız

ChartIndicatorGet()

IndicatorRelease

Gösterge tanıtıcı değerini kaldırır ve başka biri tarafından kullanılmıyorsa hesaplama bloğunu serbest bırakır.

```
bool  IndicatorRelease(
   int       indicator_handle     // gösterge tanıtıcı değeri
   );
```

Dönüş değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Not

Fonksiyon, göstergenin artık ihtiyaç olmayan tanıtıcı değerini kaldırır. Böylece bellekten tasarruf edilir. İşleyici hemen, hesaplama bloğu ise bir süre sonra kaldırılır (artık çağrılmıyorsa).

Strateji sınayıcı ile çalışırken, IndicatorRelease() fonksiyonu çalıştırılmaz.

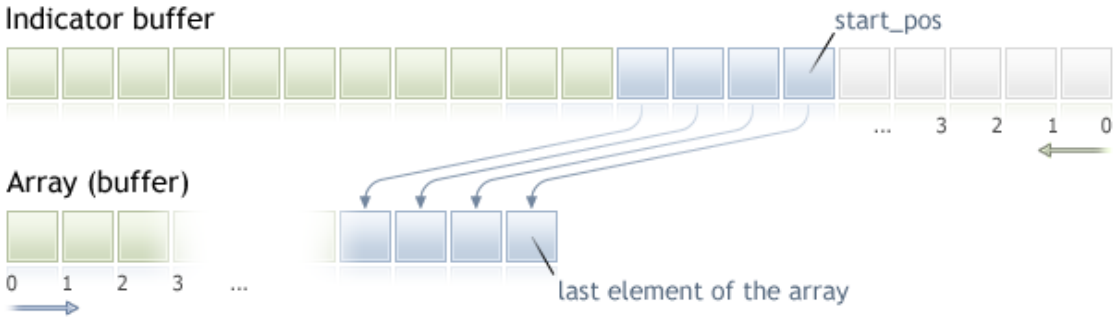
Örnek:

```
//+------------------------------------------------------------------+
//|                                      Test_IndicatorRelease.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "2010, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- giriş parametreleri
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;
//--- gösterge tanıtıcı değerini saklayacağız
int MA_handle=INVALID_HANDLE;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tanıtıcı değerini oluştur
   MA_handle=iMA(Symbol(),0,MA_Period,MA_shift,MA_smooth,PRICE_CLOSE);
//--- global değişkenleri sil
   if(GlobalVariableCheck("MA_value"))
      GlobalVariableDel("MA_value");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
```

```
//+------------------------------------------------------------------+
void OnTick()
  {
//--- global değişkenin değeri mevcut değilse
   if(!GlobalVariableCheck("MA_value"))
     {
      //--- son iki çubuktaki gösterge değerlerini al
      if(MA_handle!=INVALID_HANDLE)
        {
         //--- gösterge değerleri için bir dinamik dizi
         double values[];
         if(CopyBuffer(MA_handle,0,0,2,values)==2 && values[0]!=EMPTY_VALUE)
           {
            //--- son çubuk için global değişkenin değerini hatırla
            if(GlobalVariableSet("MA_value",values[0]))
              {
               //--- gösterge tanıtıcı değerini serbest bırak
               if(!IndicatorRelease(MA_handle))
                  Print("IndicatorRelease() başarısız oldu. Hata",GetLastError());
               else MA_handle=INVALID_HANDLE;
              }
            else
               Print("GlobalVariableSet başarısız oldu. Hata",GetLastError());
           }
        }
     }
//---
  }
```

CopyBuffer

Belirtilen bir göstergenin belirtilen bir tamponundan gereken miktarda veri alır.



Kopyalanan verinin elemanlarının (buffer_num indisine sahip tamponu) sayılması şimdiden geçmişe doğru gerçekleşir, yani 0 başlangıç konumu mevcut çubuk anlamına gelir.

Bilinmeyen miktardaki bir veriyi kopyalarken, buffer[] alıcı tamponu için dinamik dizilerin kullanılması önerilir; çünkü CopyBuffer() fonksiyonu, alıcı diziye, kopyalanan verinin büyüklüğü kadar bellek tahsis etmeye çalışacaktır. Eğer bir gösterge tamponu (SetIndexBufer() fonksiyonu ile değerlerini saklamak için önceden tahsis edilen dizi), buffer[] alıcı dizisi olarak kullanılıyorsa, kısmi kopyalamaya izin verilir. Bu duruma bir örnek, terminal standart paketindeki Awesome_Oscillator.mql5 özel göstergesinde bulunabilir.

Bir gösterge tamponunun değerlerini kısmen başka bir diziye kopyalamak istiyorsanız (gösterge tamponu olmayan bir diziye), istenen verinin kopyalanacağı bir ara dizi kullanmalısınız. Ardından, istenen sayıdaki değeri, alıcı dizinin istenen yerlerine elementsel olarak kopyalayın.

Eğer kopyalayacağınız verinin tam miktarını biliyorsanız, aşırı bellek tahsisini önlemek amacıyla statik olarak tahsis edilmiş bir tampon kullanmanız daha iyi olacaktır.

Hedef dizisinin erişim özelliğinin ne olduğu önem taşımaz - as_series=true veya as_series=false. Veri, en eski elemanın tahsis edilen fiziksel belleğin başlangıcında konumlanacağı şekilde yerleştirilir. Bu fonksiyonun 3 çağrı şekli bulunmaktadır.

İlk pozisyon ve istenen eleman sayısı ile çağrı

```
int  CopyBuffer(
   int       indicator_handle,     // gösterge tanıtıcı değeri
   int       buffer_num,           // gösterge tamponunun numarası
   int       start_pos,            // başlangıç pozisyonu
   int       count,                // kopyalanacak veri miktarı
   double    buffer[]              // hedef dizi
   );
```

Başlangıç tarihi ve istenen eleman sayısı ile çağrı

```
int  CopyBuffer(
   int       indicator_handle,     // gösterge tanıtıcı değeri
   int       buffer_num,           // gösterge tamponunun numarası
   datetime  start_time,           // başlangıç tarihi ve zamanı
```

```
   int       count,                // kopyalanacak veri miktarı
   double    buffer[]              // hedef dizi
   );
```

İstene aralığın başlangıç ve bitiş tarihi ile çağrı

```
int  CopyBuffer(
   int       indicator_handle,     // gösterge tanıtıcı değeri
   int       buffer_num,           // gösterge tamponunun numarası
   datetime  start_time,           // başlangıç tarihi ve zamanı
   datetime  stop_time,            // son tarih ve zaman
   double    buffer[]              // hedef dizi
   );
```

Parametreler

indicator_handle

[in] Karşılık gelen fonksiyonla dönüşü yapılan gösterge tanıtıcı değeri.

buffer_num

[in] Gösterge tamponunun numarası.

start_pos

[in] Kopyalanacak ilk elemanın konumu.

count

[in] Kopyalanacak veri miktarı.

start_time

[in] İlk elemana denk gelen çubuk zamanı.

stop_time

[in] Son elemana denk gelen çubuk zamanı.

buffer[]

[out] double tipli dizi.

Dönüş değeri

Kopyalanan veri sayısına veya hata durumunda -1 değerine dönüş yapar.

Not

Bir göstergeden veri istenirken, istenen zaman-serisi henüz kurulmamışsa veya sunucudan yüklenmemişse, fonksiyon hemen -1 dönüşü yapacaktır ve yükleme/kurma işlemi başlatılacaktır.

Bir Uzman Danışmandan veya bir scriptten veri istenirken, istenen veri terminalde yerel olarak bulunmuyorsa sunucudan yükleme işlemi başlatılacaktır, eğer veri kurulu durumdaysa ama hazır değilse, o zaman serinin kurulumuna başlanacaktır. Varsayılan zaman-aşımı süresi geçildiğinde, fonksiyon hazır olan verinin miktarına dönüş yapacaktır.

Örnek:

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

```
//|                                                TestCopyBuffer3.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                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
//--- giriş parametreleri
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;
//--- gösterge tamponları
double                   MABuffer[];
int                      ma_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("Parametre AsSeries = ",AsSeries);
   Print("SetIndexBuffer() çağrısından sonra gösterge tamponu bir zaman-serisi = ",
         ArrayGetAsSeries(MABuffer));
//--- gösterge kısa ismini ayarla
   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//--- AsSeries(giriş parametrelerine bağlı olarak) çağrısını ayarla
   ArraySetAsSeries(MABuffer,AsSeries);
   Print("ArraySetAsSeries(MABuffer,true) çağrısından sonra gösterge tamponu bir zaman
         ArrayGetAsSeries(MABuffer));
//---
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
```

```
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[])
  {
//--- tüm veriler hesaplanmış mı kontrol et
   if(BarsCalculated(ma_handle)<rates_total) return(0);
//--- tüm verileri kopyalayamayız
   int to_copy;
   if(prev_calculated>rates_total || prev_calculated<=0) to_copy=rates_total;
   else
     {
      to_copy=rates_total-prev_calculated;
      //--- son değer her zaman kopyalanır
      to_copy++;
     }
//--- kopyalamaya çalış
   if(CopyBuffer(ma_handle,0,0,to_copy,MABuffer)<=0) return(0);
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

Yukarıdaki örnekte, bir gösterge tamponunun aynı sembol/periyot değerine sahip bir göstergenin tamponuyla nasıl doldurulacağı gösterilmektedir.

Tarihsel veri istemiyle ilgili detaylı bir örnek için Nesne Bağlama Yöntemleri bölümüne bakınız. Söz konusu örnekte iFractals göstergesinden son 1000 alınması ve son 10 yukarı/aşağı fraktalin çizelge üzerine çizilmesi gösterilmektedir. Benzer bir teknik, eksik veriye sahip olan ve genellikle aşağıda belirtilen stillerle çizilen tüm göstergelerde kullanılabilir:

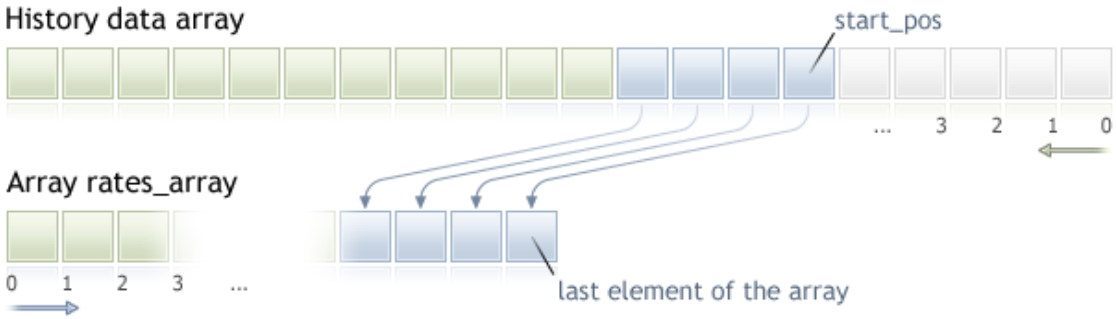
- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

Ayrıca Bakınız

Özel Göstergelerin Özellikleri, SetIndexBuffer

CopyRates

Belirtilen sembol ve periyot için MqlRates yapısının tarihsel verisini rates_array dizisine belirtilen miktarda kopyalar. Kopyalanan verinin elemanlarının sıralanma işlemi, mevcut zamandan geçmişe doğru gerçekleşir; yani 0 başlangıç konumu mevcut çubuk anlamına gelir.



Bilinmeyen miktardaki bir veriyi kopyalarken, hedef dizi olarak bir dinamik dizinin kullanılması önerilir. Çünkü istenen veri miktarı hedef dizinin büyüklüğünden azsa (veya fazlaysa), fonksiyon, verinin tam olarak uyması için yeni bellek tahsisi yapacaktır.

Eğer kopyalayacağınız verinin tam miktarını biliyorsanız, aşırı bellek tahsisini önlemek amacıyla statik olarak tahsis edilmiş bir tampon kullanmanız daha iyi olacaktır.

Hedef dizisinin erişim özelliğinin ne olduğu önem taşımaz - as_series=true veya as_series=false. Veri, en eski elemanın tahsis edilen fiziksel belleğin başlangıcında konumlanacağı şekilde yerleştirilir. Bu fonksiyonun 3 çağrı şekli bulunmaktadır.

İlk pozisyon ve istenen eleman sayısı ile çağrı

```
int  CopyRates(
   string           symbol_name,       // sembol ismi
   ENUM_TIMEFRAMES  timeframe,         // periyot
   int              start_pos,         // başlangıç konumu
   int              count,             // kopyalanacak veri miktarı
   MqlRates         rates_array[]      // hedef dizi
   );
```

Başlangıç tarihi ve istenen eleman sayısı ile çağrı

```
int  CopyRates(
   string           symbol_name,       // sembol ismi
   ENUM_TIMEFRAMES  timeframe,         // periyot
   datetime         start_time,        // başlangıç tarihi ve zamanı
   int              count,             // kopyalanacak veri miktarı
   MqlRates         rates_array[]      // hedef dizi
   );
```

İstene aralığın başlangıç ve bitiş tarihi ile çağrı

```
int  CopyRates(
```

```
   string           symbol_name,       // sembol ismi
   ENUM_TIMEFRAMES  timeframe,         // periyot
   datetime         start_time,        // başlangıç tarihi ve zamanı
   datetime         stop_time,         // bitiş tarihi ve zamanı
   MqlRates         rates_array[]      // hedef dizi
   );
```

Parametreler

symbol_name

[in] Sembol ismi.

timeframe

[in] Periyot.

start_time

[in] Kopyalanacak ilk eleman için başlangıç zamanı.

start_pos

[in] Kopyalanacak ilk eleman için başlangıç konumu.

count

[in] Kopyalanacak veri miktarı.

stop_time

[in] Kopyalanacak son elemana karşılık gelen çubuğun zamanı.

rates_array[]

[out] MqlRates tipli bir dizi.

Dönüş değeri

Kopyalanan veri sayısına veya hata durumunda -1 değerine dönüş yapar.

Not

İstenen veri aralığı sunucuda bulunan mevcut veri aralığının dışındaysa, fonksiyon -1 dönüşü yapar. Eğer istenen veri miktarı TERMINAL_MAXBARS değerinin üstündeyse (çizelgedeki maksimum çubuk sayısı), fonksiyon yine -1 dönüşü yapacaktır.

Bir göstergeden veri istenirken, istenen zaman-serisi henüz kurulmamışsa veya sunucudan yüklenmemişse, fonksiyon hemen -1 dönüşü yapacaktır ve yükleme/kurma işlemi başlatılacaktır.

Bir Uzman Danışmandan veya bir scriptten veri istenirken, istenen veri terminalde yerel olarak bulunmuyorsa sunucudan yükleme işlemi başlatılacaktır, eğer veri kurulu durumdaysa ama hazır değilse, o zaman serinin kurulumuna başlanacaktır. Varsayılan zaman-aşımı süresi geçildiğinde, fonksiyon hazır olan verinin miktarına dönüş yapacaktır ama geçmiş yüklemesi devam edecektir ve bir sonraki benzer istekte, fonksiyon daha fazla veri dönüşü yapacaktır.

Başlangıç tarihi ve istenen veri miktarını kullanarak bir istek gerçekleştirildiğinde, sadece belirtilen tarihe eşit veya ondan daha önceki verilerin dönüşü yapılacaktır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman belirtilenden az veya belirtilene eşit olacaktır.

Belirtilen tarih aralığındaki bir veri istendiğinde, sadece bu aralıktaki verilere dönüş yapılır. Zaman aralığı ayarlanır ve saniyeye kadar sayılır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman istenilen aralık içindedir.

Bu şekilde, *start_time=Last_Tuesday* ve *stop_time=Last_Friday* aralığındaki bir haftalık verinin kopyalanması isteniyorsa ve mevcut gün cumartesi ise, fonksiyon 0 dönüşü yapacaktır. Çünkü bir haftalık zaman aralığında açılış günü her zaman Pazar günüdür ve bir haftalık çubuk belirtilen aralığa düşmez.

Eğer tamamlanmamış mevcut çubuğun verisine dönüş yaptırmak istiyorsanız, *start_pos*=0 ve *count*=1 değerlerini ilk çağrı versiyonunda kullanabilirsiniz.

Örnek:

```
void OnStart()
  {
//---
   MqlRates rates[];
   ArraySetAsSeries(rates,true);
   int copied=CopyRates(Symbol(),0,0,100,rates);
   if(copied>0)
     {
      Print("Kopyalanan çubuklar: "+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(Symbol()," sembolü için tarihsel veri alınamadı");
  }
```

Tarihsel veri istemiyle ilgili detaylı bir örnek için Nesne Bağlama Yöntemleri bölümüne bakınız. Söz konusu örnekte iFractals göstergesinden son 1000 alınması ve son 10 yukarı/aşağı fraktalin çizelge üzerine çizilmesi gösterilmektedir. Benzer bir teknik, eksik veriye sahip olan ve genellikle aşağıda belirtilen stillerle çizilen tüm göstergelerde kullanılabilir:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,

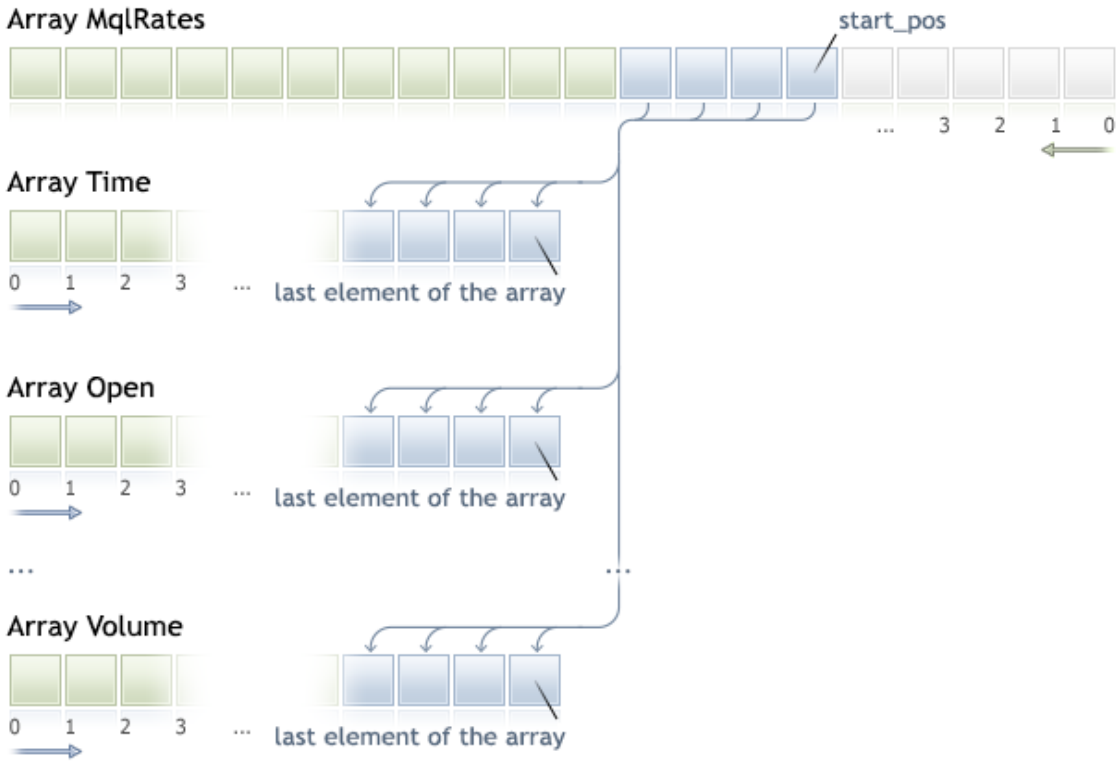
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

Ayrıca Bakınız

Yapılar ve Sınıflar, TimeToString, StringFormat

CopySeries

Belirtilen sembol, zaman dilimi ve veri miktarı için MqlRates yapısından belirtilen dizi kümesine senkronize zaman serilerini alır. Elemanlar şimdiki zamandan geçmişe doğru sayılır, yani 0'a eşit başlangıç konumu mevcut çubuk anlamına gelir.



Kopyalanacak veri miktarı bilinmiyorsa, alıcı diziler için dinamik dizi kullanılması önerilir. Çünkü eğer kopyalanan veri miktarı dizinin içerebileceğinden daha fazla olursa, talep edilen tüm verilerin diziye sığması amacıyla dizide yeniden dağıtım gerçekleşebilir.

Belirli bir miktarda veri kopyalanacaksa, gereksiz bellek yeniden tahsisini önlemek adına statik olarak tahsis edilmiş arabellek kullanılması önerilir.

Alıcı dizinin as_series=true veya as_series=false olmak üzere hangi özelliğe sahip olduğu önemli değildir: veriler, dizi için tahsis edilmiş fiziksel belleğin başına en eski zaman serisi elemanı gelecek şekilde kopyalanacaktır.

```
int  CopySeries(
   string           symbol_name,       // sembolün adı
   ENUM_TIMEFRAMES  timeframe,         // zaman dilimi
   int              start_pos,         // başlangıç konumu
   int              count,             // kopyalanacak veri miktarı
   ulong            rates_mask,        // istenen zaman serilerini belirtmek için bay
   void&            array1[],          // ilk zaman serisinin verilerinin kopyalanaca
   void&            array2[]           // ikinci zaman serisinin verilerinin kopyalan
   ...
   );
```

Parametreler

symbol_name

[in] Sembol.

timeframe

[in] Zaman dilimi.

start_pos

[in] Kopyalanacak ilk elemanın indeksi.

count

[in] Kopyalanacak eleman sayısı.

rates_mask

[in] ENUM_COPY_RATES numaralandırmasından bayrak kombinasyonu.

array1, array2,...

[out] MqlRates yapısından zaman serisini almak için uygun türde dizi. Fonksiyona iletilen dizilerin sırası, MqlRates yapısındaki alanların sırasıyla eşleşmelidir.

Geri dönüş değeri

Kopyalanan elemanların sayısı veya hata durumunda -1.

Not

Talep edilen veri aralığı sunucuda bulunan verileri aşıyorsa, fonksiyon -1 geri döndürür. Talep edilen veri aralığı TERMINAL_MAXBARS'ı (grafikteki maksimum çubuk sayısı) aşıyorsa, fonksiyon yine -1 geri döndürür.

Göstergeden veri talep edilirken, istenen zaman serileri henüz oluşturulmadıysa veya sunucudan indirilmesi gerekiyorsa, fonksiyon hemen -1 geri döndürür. Böyle bir durumda, aynı zamanda verilerin indirilmesi/oluşturulması da başlatılır.

Uzman Danışmandan veya komut dosyasından veri talep edilirken, terminalde yerel olarak ilgili veriler yoksa, sunucudan indirme işlemi başlatılır ve bu süreçte yerel geçmişten oluşturulabiliyorsa gerekli zaman serileri oluşturulur. Fonksiyon, zaman aşımı süresi dolduğunda hazır olan veri miktarını geri döndürür, ancak geçmişin indirilme işlemi halen devam eder, böylece bir sonraki benzer talepte daha fazla veri geri döndürülür.

CopySeries ve CopyRates arasındaki fark

CopySeries fonksiyonu, tek bir çağrıyla yalnızca gerekli zaman serilerinin farklı dizilere alınmasına olanak sağlar ve bu sırada zaman serilerinin tamamı senkronize edilir. Bu, belirli bir N indeksi için elde edilen dizilerdeki tüm değerlerin belirtilen sembol / zaman dilimindeki aynı çubuğa ait olacağı anlamına gelir. Böylece, programcının alınan tüm zaman serilerini çubuğun açılış zamanına göre senkronize etmesine gerek kalmaz.

Zaman serilerinin tamamını bir MqlRates dizisi olarak geri döndüren CopyRates'in aksine, CopySeries fonksiyonu yalnızca gerekli zaman serilerinin farklı diziler halinde alınmasını mümkün kılar. Bu, istenen

zaman serilerini belirtmek üzere bayrak kombinasyonu ayarlanarak yapılır. Fonksiyona iletilen dizilerin sırası, MqlRates yapısındaki alanların sırasıyla eşleşmelidir:

```
struct MqlRates
  {
   datetime time;         // açılış zamanı
   double   open;         // açılış fiyatı
   double   high;         // yüksek fiyatı
   double   low;          // düşük fiyatı
   double   close;        // kapanış fiyatı
   long     tick_volume;  // tik hacmi
   int      spread;       // spread
   long     real_volume;  // gerçek hacim
  }
```

Örneğin, mevcut sembol / zaman diliminde son 100 çubuk için *time*, *close* ve *real_volume* zaman serilerinin değerleri alınması gerekiyorsa, çağrı aşağıdaki gibi olmalıdır:

```
datetime time[];
double   close[];
long     volume[];
CopySeries(NULL,0,0,100,COPY_RATES_TIME|COPY_RATES_CLOSE|COPY_RATES_VOLUME_REAL,time,c
```

Burada, *time*, *close* ve *volume* dizilerinin sırası önemlidir: MqlRates yapısındaki alanların sırasına karşılık gelmelidir. Ancak *rates_mask* içerisindeki değerlerin sırası önemli değildir. Maske şu şekilde olabilir:

```
COPY_RATES_VOLUME_REAL|COPY_RATES_TIME|COPY_RATES_CLOSE
```

Örnek:

```
//--- girdi parametreleri
input datetime InpDateFrom=D'2022.01.01 00:00:00';
input datetime InpDateTo  =D'2023.01.01 00:00:00';
input uint     InpCount   =20;
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
void OnStart(void)
  {
//--- MqlRates yapısından zaman serileri almak için diziler
   double   open[];
   double   close[];
   float    closef[];
   datetime time1[], time2[];
//--- double türünde diziye kapanış fiyatlarını talep et
   ResetLastError();
   int res1=CopySeries(NULL, PERIOD_CURRENT, 0, InpCount,
                       COPY_RATES_TIME|COPY_RATES_CLOSE, time1, close);
   PrintFormat("1. CopySeries %d değer geri döndürdü. Hata kodu=%d", res1, GetLastErro
```

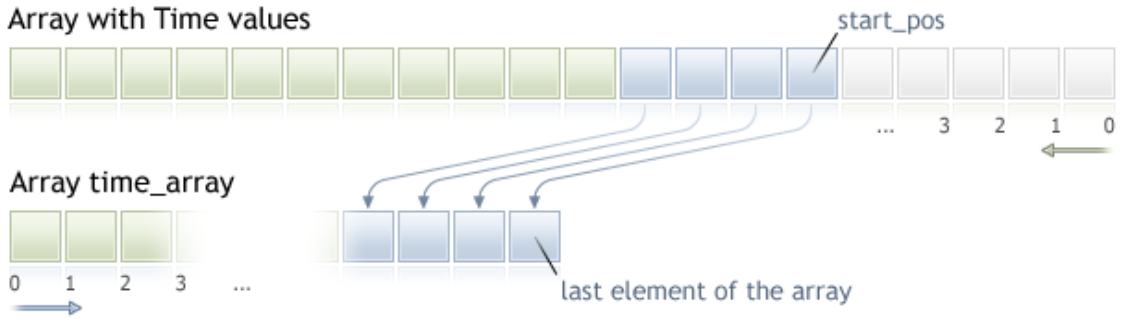
```
   ArrayPrint(close);

//--- açılış fiyatlarını da talep et ve kapanış fiyatları için float türünü kullan
   ResetLastError();
   int res2=CopySeries(NULL, PERIOD_CURRENT, 0, InpCount,
                       COPY_RATES_TIME|COPY_RATES_CLOSE|COPY_RATES_OPEN, time2, open,
   PrintFormat("2. CopySeries %d değer geri döndürdü. Hata kodu=%d", res2, GetLastErro
   ArrayPrint(closef);
//--- alınan verileri karşılaştır
   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]);
        }
     }
//--- Sonuç
      1. CopySeries 20 değer geri döndürdü. Hata kodu:0 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 20 değer geri döndürdü. Hata kodu:0 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   |
//---
  }
```

Ayrıca bakınız

Yapılar ve sınıflar, CopyRates

CopyTime

Seçilen sembol-periyot çifti için, geçmiş çubukların açılış zamanı verilerini time_array dizisinin içine belirtilen miktarda kopyalar. Elemanların sıralanma şeklinin şimdiden geçmişe doğru olduğu not edilmelidir, yani 0 başlangıç konumu mevcut çubuk anlamına gelecektir.



Bilinmeyen miktardaki bir veriyi kopyalarken, hedef dizi olarak bir dinamik dizinin kullanılması önerilir. Çünkü istenen veri miktarı hedef dizinin büyüklüğünden azsa (veya fazlaysa), fonksiyon, verinin tam olarak uyması için yeni bellek tahsisi yapacaktır.

Eğer kopyalayacağınız verinin tam miktarını biliyorsanız, aşırı bellek tahsisini önlemek amacıyla statik olarak tahsis edilmiş bir tampon kullanmanız daha iyi olacaktır.

Hedef dizisinin erişim özelliğinin ne olduğu önem taşımaz - as_series=true veya as_series=false. Veri, en eski elemanın tahsis edilen fiziksel belleğin başlangıcında konumlanacağı şekilde yerleştirilir. Bu fonksiyonun 3 çağrı şekli bulunmaktadır.

İlk pozisyon ve istenen eleman sayısı ile çağrı

```
int  CopyTime(
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   int              start_pos,        // kopyalanacak veri miktarı
   int              count,            // kopyalanacak veri miktarı
   datetime         time_array[]      // açılış fiyatları için hedef dizi
   );
```

Başlangıç tarihi ve istenen eleman sayısı ile çağrı

```
int  CopyTime(
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   datetime         start_time,       // başlangıç tarihi ve zamanı
   int              count,            // kopyalanacak veri miktarı
   datetime         time_array[]      // açılış fiyatları için hedef dizi
   );
```

İstene aralığın başlangıç ve bitiş tarihi ile çağrı

```
int  CopyTime(
```

```
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   datetime         start_time,       // başlangıç tarihi ve zamanı
   datetime         stop_time,        // bitiş tarihi ve zamanı
   datetime         time_array[]      // açılış fiyatları için hedef dizi
   );
```

Parametreler

symbol_name

[in] Sembol ismi.

timeframe

[in] Periyot.

start_pos

[in] Kopyalanacak ilk eleman için başlangıç konumu.

count

[in] Kopyalanacak veri miktarı.

start_time

[in] Kopyalanacak ilk eleman için başlangıç zamanı.

stop_time

[in] Kopyalanacak son elemana karşılık gelen çubuğun zamanı.

time_array[]

[out] datetime tipli dizi.

Dönüş değeri

Kopyalanan veri sayısına veya hata durumunda -1 değerine dönüş yapar.

Not

İstenen veri aralığı sunucuda bulunan mevcut veri aralığının dışındaysa, fonksiyon -1 dönüşü yapar. Eğer istenen veri miktarı TERMINAL_MAXBARS değerinin üstündeyse (çizelgedeki maksimum çubuk sayısı), fonksiyon yine -1 dönüşü yapacaktır.

Bir göstergeden veri istenirken, istenen zaman-serisi henüz kurulmamışsa veya sunucudan yüklenmemişse, fonksiyon hemen -1 dönüşü yapacaktır ve yükleme/kurma işlemi başlatılacaktır.

Bir Uzman Danışmandan veya bir scriptten veri istenirken, istenen veri terminalde yerel olarak bulunmuyorsa sunucudan yükleme işlemi başlatılacaktır, eğer veri kurulu durumdaysa ama hazır değilse, o zaman serinin kurulumuna başlanacaktır. Varsayılan zaman-aşımı süresi geçildiğinde, fonksiyon hazır olan verinin miktarına dönüş yapacaktır ama geçmiş yüklemesi devam edecektir ve bir sonraki benzer istekte, fonksiyon daha fazla veri dönüşü yapacaktır.

Başlangıç tarihi ve istenen veri miktarını kullanarak bir istek gerçekleştirildiğinde, sadece belirtilen tarihe eşit veya ondan daha önceki verilerin dönüşü yapılacaktır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman belirtilenden az veya belirtilene eşit olacaktır.

Belirtilen tarih aralığındaki bir veri istendiğinde, sadece bu aralıktaki verilere dönüş yapılır. Zaman aralığı ayarlanır ve saniyeye kadar sayılır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman istenilen aralık içindedir.

Bu şekilde, *start_time=Last_Tuesday* ve *stop_time=Last_Friday* aralığındaki bir haftalık verinin kopyalanması isteniyorsa ve mevcut gün cumartesi ise, fonksiyon 0 dönüşü yapacaktır. Çünkü bir haftalık zaman aralığında açılış günü her zaman Pazar günüdür ve bir haftalık çubuk belirtilen aralığa düşmez.

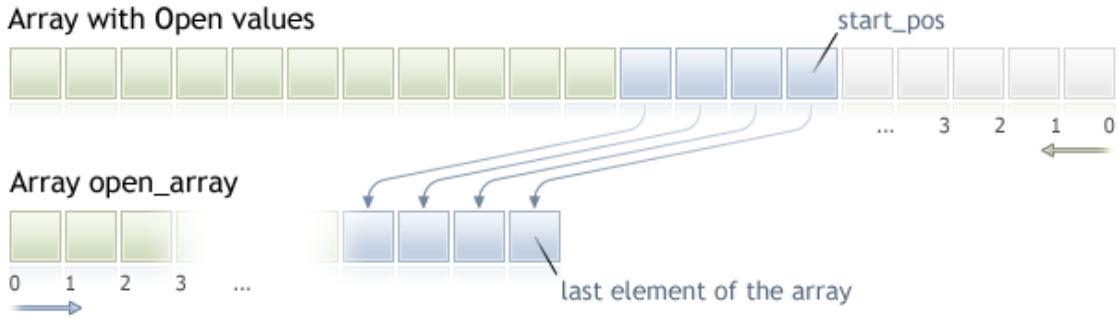
Eğer tamamlanmamış mevcut çubuğun verisine dönüş yaptırmak istiyorsanız, *start_pos*=0 ve *count*=1 değerlerini ilk çağrı versiyonunda kullanabilirsiniz.

Tarihsel veri istemiyle ilgili detaylı bir örnek için Nesne Bağlama Yöntemleri bölümüne bakınız. Söz konusu örnekte iFractals göstergesinden son 1000 alınması ve son 10 yukarı/aşağı fraktalin çizelge üzerine çizilmesi gösterilmektedir. Benzer bir teknik, eksik veriye sahip olan ve genellikle aşağıda belirtilen stillerle çizilen tüm göstergelerde kullanılabilir:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

CopyOpen

Seçilen sembol-periyot çifti için, geçmiş açılış fiyatı verilerini open_array dizisinin içine belirtilen miktarda kopyalar. Elemanların sıralanma şeklinin şimdiden geçmişe doğru olduğu not edilmelidir, yani 0 başlangıç konumu mevcut çubuk anlamına gelecektir.



Bilinmeyen miktardaki bir veriyi kopyalarken, hedef dizi olarak bir dinamik dizinin kullanılması önerilir. Çünkü istenen veri miktarı hedef dizinin büyüklüğünden azsa (veya fazlaysa), fonksiyon, verinin tam olarak uyması için yeni bellek tahsisi yapacaktır.

Eğer kopyalayacağınız verinin tam miktarını biliyorsanız, aşırı bellek tahsisini önlemek amacıyla statik olarak tahsis edilmiş bir tampon kullanmanız daha iyi olacaktır.

Hedef dizisinin erişim özelliğinin ne olduğu önem taşımaz - as_series=true veya as_series=false. Veri, en eski elemanın tahsis edilen fiziksel belleğin başlangıcında konumlanacağı şekilde yerleştirilir. Bu fonksiyonun 3 çağrı şekli bulunmaktadır.

İlk pozisyon ve istenen eleman sayısı ile çağrı

```
int  CopyOpen(
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   int              start_pos,        // kopyalanacak veri miktarı
   int              count,            // kopyalanacak veri miktarı
   double           open_array[]      // hedef dizi
   );
```

Başlangıç tarihi ve istenen eleman sayısı ile çağrı

```
int  CopyOpen(
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   datetime         start_time,       // başlangıç tarihi ve zamanı
   int              count,            // kopyalanacak veri miktarı
   double           open_array[]      // açılış fiyatları için hedef dizi
   );
```

İstene aralığın başlangıç ve bitiş tarihi ile çağrı

```
int  CopyOpen(
```

```
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   datetime         start_time,       // başlangıç tarihi ve zamanı
   datetime         stop_time,        // bitiş tarihi ve zamanı
   double           open_array[]      // açılış fiyatları için hedef dizi
   );
```

Parametreler

symbol_name

[in] Sembol ismi.

timeframe

[in] Periyot.

start_pos

[in] Kopyalanacak ilk eleman için başlangıç konumu.

count

[in] Kopyalanacak veri miktarı.

start_time

[in] Kopyalanacak ilk eleman için başlangıç zamanı.

stop_time

[in] Kopyalanacak son eleman için başlangıç zamanı.

open_array[]

[out] double tipli dizi.

Dönüş değeri

Kopyalanan veri sayısına veya hata durumunda -1 değerine dönüş yapar.

Not

İstenen veri aralığı sunucuda bulunan mevcut veri aralığının dışındaysa, fonksiyon -1 dönüşü yapar. Eğer istenen veri miktarı TERMINAL_MAXBARS değerinin üstündeyse (çizelgedeki maksimum çubuk sayısı), fonksiyon yine -1 dönüşü yapacaktır.

Bir göstergeden veri istenirken, istenen zaman-serisi henüz kurulmamışsa veya sunucudan yüklenmemişse, fonksiyon hemen -1 dönüşü yapacaktır ve yükleme/kurma işlemi başlatılacaktır.

Bir Uzman Danışmandan veya bir scriptten veri istenirken, istenen veri terminalde yerel olarak bulunmuyorsa sunucudan yükleme işlemi başlatılacaktır, eğer veri kurulu durumdaysa ama hazır değilse, o zaman serinin kurulumuna başlanacaktır. Varsayılan zaman-aşımı süresi geçildiğinde, fonksiyon hazır olan verinin miktarına dönüş yapacaktır ama geçmiş yüklemesi devam edecektir ve bir sonraki benzer istekte, fonksiyon daha fazla veri dönüşü yapacaktır.

Başlangıç tarihi ve istenen veri miktarını kullanarak bir istek gerçekleştirildiğinde, sadece belirtilen tarihe eşit veya ondan daha önceki verilerin dönüşü yapılacaktır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman belirtilenden az veya belirtilene eşit olacaktır.

Belirtilen tarih aralığındaki bir veri istendiğinde, sadece bu aralıktaki verilere dönüş yapılır. Zaman aralığı ayarlanır ve saniyeye kadar sayılır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman istenilen aralık içindedir.

Bu şekilde, *start_time=Last_Tuesday* ve *stop_time=Last_Friday* aralığındaki bir haftalık verinin kopyalanması isteniyorsa ve mevcut gün cumartesi ise, fonksiyon 0 dönüşü yapacaktır. Çünkü bir haftalık zaman aralığında açılış günü her zaman Pazar günüdür ve bir haftalık çubuk belirtilen aralığa düşmez.

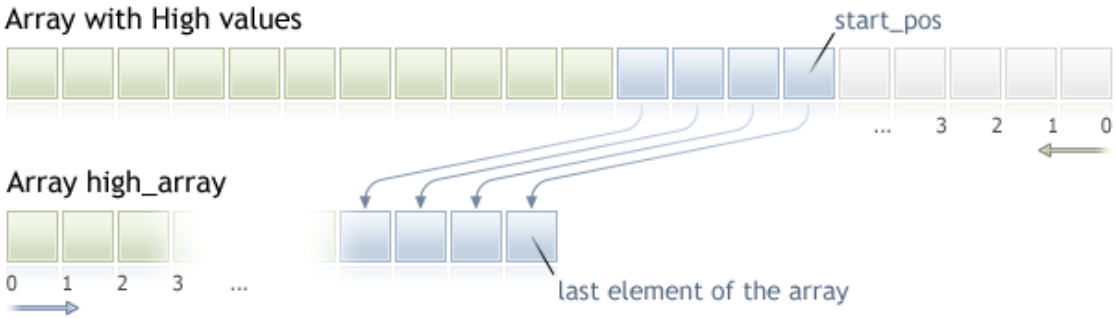
Eğer tamamlanmamış mevcut çubuğun verisine dönüş yaptırmak istiyorsanız, *start_pos*=0 ve *count*=1 değerlerini ilk çağrı versiyonunda kullanabilirsiniz.

Tarihsel veri istemiyle ilgili detaylı bir örnek için Nesne Bağlama Yöntemleri bölümüne bakınız. Söz konusu örnekte iFractals göstergesinden son 1000 alınması ve son 10 yukarı/aşağı fraktalin çizelge üzerine çizilmesi gösterilmektedir. Benzer bir teknik, eksik veriye sahip olan ve genellikle aşağıda belirtilen stillerle çizilen tüm göstergelerde kullanılabilir:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

CopyHigh

Bu fonksiyon, seçilen sembol-periyot çifti için, geçmiş en yüksek fiyat verilerini high_array dizisinin içine belirtilen miktarda kopyalar. Elemanların sıralanma şeklinin şimdiden geçmişe doğru olduğu not edilmelidir, yani 0 başlangıç konumu mevcut çubuk anlamına gelecektir.



Bilinmeyen miktardaki bir veriyi kopyalarken, hedef dizi olarak bir dinamik dizinin kullanılması önerilir. Çünkü istenen veri miktarı hedef dizinin büyüklüğünden azsa (veya fazlaysa), fonksiyon, verinin tam olarak uyması için yeni bellek tahsisi yapacaktır.

Eğer kopyalayacağınız verinin tam miktarını biliyorsanız, aşırı bellek tahsisini önlemek amacıyla statik olarak tahsis edilmiş bir tampon kullanmanız daha iyi olacaktır.

Hedef dizisinin erişim özelliğinin ne olduğu önem taşımaz - as_series=true veya as_series=false. Veri, en eski elemanın tahsis edilen fiziksel belleğin başlangıcında konumlanacağı şekilde yerleştirilir. Bu fonksiyonun 3 çağrı şekli bulunmaktadır.

İlk pozisyon ve istenen eleman sayısı ile çağrı

```
int  CopyHigh(
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   int              start_pos,        // başlangıç konumu
   int              count,            // kopyalanacak veri miktarı
   double           high_array[]      // hedef dizi
   );
```

Başlangıç tarihi ve istenen eleman sayısı ile çağrı

```
int  CopyHigh(
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   datetime         start_time,       // başlangıç tarihi ve zamanı
   int              count,            // kopyalanacak veri miktarı
   double           high_array[]      // hedef dizi
   );
```

İstene aralığın başlangıç ve bitiş tarihi ile çağrı

```
int  CopyHigh(
```

```
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   datetime         start_time,       // başlangıç tarihi ve zamanı
   datetime         stop_time,        // bitiş tarihi ve zamanı
   double           high_array[]      // hedef dizi
   );
```

Parametreler

symbol_name

[in] Sembol ismi.

timeframe

[in] Periyot.

start_pos

[in] Kopyalanacak ilk eleman için başlangıç konumu.

count

[in] Kopyalanacak veri miktarı.

start_time

[in] Kopyalanacak ilk eleman için başlangıç zamanı.

stop_time

[in] Kopyalanacak son elemana karşılık gelen çubuğun zamanı.

high_array[]

[out] double tipli dizi.

Dönüş değeri

Kopyalanan veri sayısına veya hata durumunda -1 değerine dönüş yapar.

Not

İstenen veri aralığı sunucuda bulunan mevcut veri aralığının dışındaysa, fonksiyon -1 dönüşü yapar. Eğer istenen veri miktarı TERMINAL_MAXBARS değerinin üstündeyse (çizelgedeki maksimum çubuk sayısı), fonksiyon yine -1 dönüşü yapacaktır.

Bir göstergeden veri istenirken, istenen zaman-serisi henüz kurulmamışsa veya sunucudan yüklenmemişse, fonksiyon hemen -1 dönüşü yapacaktır ve yükleme/kurma işlemi başlatılacaktır.

Bir Uzman Danışmandan veya bir scriptten veri istenirken, istenen veri terminalde yerel olarak bulunmuyorsa sunucudan yükleme işlemi başlatılacaktır, eğer veri kurulu durumdaysa ama hazır değilse, o zaman serinin kurulumuna başlanacaktır. Varsayılan zaman-aşımı süresi geçildiğinde, fonksiyon hazır olan verinin miktarına dönüş yapacaktır ama geçmiş yüklemesi devam edecektir ve bir sonraki benzer istekte, fonksiyon daha fazla veri dönüşü yapacaktır.

Başlangıç tarihi ve istenen veri miktarını kullanarak bir istek gerçekleştirildiğinde, sadece belirtilen tarihe eşit veya ondan daha önceki verilerin dönüşü yapılacaktır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman belirtilenden az veya belirtilene eşit olacaktır.

Belirtilen tarih aralığındaki bir veri istendiğinde, sadece bu aralıktaki verilere dönüş yapılır. Zaman aralığı ayarlanır ve saniyeye kadar sayılır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman istenilen aralık içindedir.

Bu şekilde, *start_time=Last_Tuesday* ve *stop_time=Last_Friday* aralığındaki bir haftalık verinin kopyalanması isteniyorsa ve mevcut gün cumartesi ise, fonksiyon 0 dönüşü yapacaktır. Çünkü bir haftalık zaman aralığında açılış günü her zaman Pazar günüdür ve bir haftalık çubuk belirtilen aralığa düşmez.

Eğer tamamlanmamış mevcut çubuğun verisine dönüş yaptırmak istiyorsanız, *start_pos*=0 ve *count*=1 değerlerini ilk çağrı versiyonunda kullanabilirsiniz.

Örnek:

```
#property copyright "2009, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "High[i] ve Low[i] çıktılarının alınmasına dair bir örnek"
#property description "seçilen rassal çubuklar için alınmasına dair bir örnek"

double High[],Low[];
//+------------------------------------------------------------------+
//| Belirtilen çubuk indisi için Low fiyatını al                     |
//+------------------------------------------------------------------+
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);
  }
//+------------------------------------------------------------------+
//| Belirtilen çubuk indisi için High fiyatını al                    |
//+------------------------------------------------------------------+
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);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
```

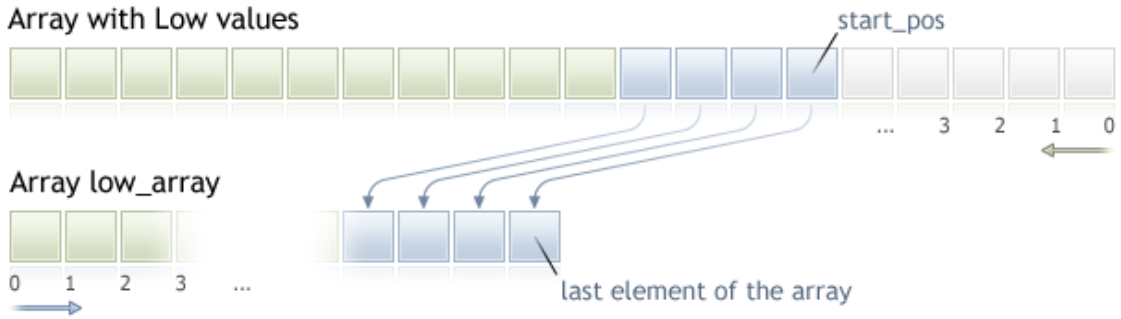
```
//--- Her tikle birlikte çubuk indisine göre High ve Low değerlerini çıktılarız,
//--- bu, tikin ulaştığı saniyeye eşittir
   datetime t=TimeCurrent();
   int sec=t%60;
   printf("High[%d] = %G  Low[%d] = %G",
          sec,iHigh(Symbol(),0,sec),
          sec,iLow(Symbol(),0,sec));
  }
```

Tarihsel veri istemiyle ilgili detaylı bir örnek için Nesne Bağlama Yöntemleri bölümüne bakınız. Söz konusu örnekte iFractals göstergesinden son 1000 alınması ve son 10 yukarı/aşağı fraktalin çizelge üzerine çizilmesi gösterilmektedir. Benzer bir teknik, eksik veriye sahip olan ve genellikle aşağıda belirtilen stillerle çizilen tüm göstergelerde kullanılabilir:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

CopyLow

Seçilen sembol-periyot çifti için, geçmiş en düşük fiyat verilerini belirtilen miktarda low_array dizisinin içine kopyalar. Elemanların sıralanma şeklinin şimdiden geçmişe doğru olduğu not edilmelidir, yani 0 başlangıç konumu mevcut çubuk anlamına gelecektir.



Bilinmeyen miktardaki bir veriyi kopyalarken, hedef dizi olarak bir dinamik dizinin kullanılması önerilir. Çünkü istenen veri miktarı hedef dizinin büyüklüğünden azsa (veya fazlaysa), fonksiyon, verinin tam olarak uyması için yeni bellek tahsisi yapacaktır.

Eğer kopyalayacağınız verinin tam miktarını biliyorsanız, aşırı bellek tahsisini önlemek amacıyla statik olarak tahsis edilmiş bir tampon kullanmanız daha iyi olacaktır.

Hedef dizisinin erişim özelliğinin ne olduğu önem taşımaz - as_series=true veya as_series=false. Veri, en eski elemanın tahsis edilen fiziksel belleğin başlangıcında konumlanacağı şekilde yerleştirilir. Bu fonksiyonun 3 çağrı şekli bulunmaktadır.

İlk pozisyon ve istenen eleman sayısı ile çağrı

```
int  CopyLow(
   string            symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES   timeframe,        // periyot
   int               start_pos,        // kopyalanacak veri miktarı
   int               count,            // kopyalanacak veri miktarı
   double            low_array[]       // hedef dizi
   );
```

Başlangıç tarihi ve istenen eleman sayısı ile çağrı

```
int  CopyLow(
   string            symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES   timeframe,        // periyot
   datetime          start_time,       // başlangıç tarihi ve zamanı
   int               count,            // kopyalanacak veri miktarı
   double            low_array[]       // hedef dizi
   );
```

İstene aralığın başlangıç ve bitiş tarihi ile çağrı

```
int  CopyLow(
```

```
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   datetime         start_time,       // başlangıç tarihi ve zamanı
   datetime         stop_time,        // bitiş tarihi ve zamanı
   double           low_array[]       // hedef dizi
   );
```

Parametreler

symbol_name

[in] Sembol.

timeframe

[in] Periyot.

start_pos

[in] Kopyalanacak ilk eleman için başlangıç konumu.

count

[in] Kopyalanacak veri miktarı.

start_time

[in] İlk elemana denk gelen çubuk zamanı.

stop_time

[in] Kopyalanacak son elemana karşılık gelen çubuğun zamanı.

low_array[]

[out] double tipli dizi.

Dönüş değeri

Kopyalanan veri sayısına veya hata durumunda -1 değerine dönüş yapar.

Not

İstenen veri aralığı sunucuda bulunan mevcut veri aralığının dışındaysa, fonksiyon -1 dönüşü yapar. Eğer istenen veri miktarı TERMINAL_MAXBARS değerinin üstündeyse (çizelgedeki maksimum çubuk sayısı), fonksiyon yine -1 dönüşü yapacaktır.

Bir göstergeden veri istenirken, istenen zaman-serisi henüz kurulmamışsa veya sunucudan yüklenmemişse, fonksiyon hemen -1 dönüşü yapacaktır ve yükleme/kurma işlemi başlatılacaktır.

Bir Uzman Danışmandan veya bir scriptten veri istenirken, istenen veri terminalde yerel olarak bulunmuyorsa sunucudan yükleme işlemi başlatılacaktır, eğer veri kurulu durumdaysa ama hazır değilse, o zaman serinin kurulumuna başlanacaktır. Varsayılan zaman-aşımı süresi geçildiğinde, fonksiyon hazır olan verinin miktarına dönüş yapacaktır ama geçmiş yüklemesi devam edecektir ve bir sonraki benzer istekte, fonksiyon daha fazla veri dönüşü yapacaktır.

Başlangıç tarihi ve istenen veri miktarını kullanarak bir istek gerçekleştirildiğinde, sadece belirtilen tarihe eşit veya ondan daha önceki verilerin dönüşü yapılacaktır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman belirtilenden az veya belirtilene eşit olacaktır.

Belirtilen tarih aralığındaki bir veri istendiğinde, sadece bu aralıktaki verilere dönüş yapılır. Zaman aralığı ayarlanır ve saniyeye kadar sayılır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman istenilen aralık içindedir.

Bu şekilde, *start_time=Last_Tuesday* ve *stop_time=Last_Friday* aralığındaki bir haftalık verinin kopyalanması isteniyorsa ve mevcut gün cumartesi ise, fonksiyon 0 dönüşü yapacaktır. Çünkü bir haftalık zaman aralığında açılış günü her zaman Pazar günüdür ve bir haftalık çubuk belirtilen aralığa düşmez.

Eğer tamamlanmamış mevcut çubuğun verisine dönüş yaptırmak istiyorsanız, *start_pos*=0 ve *count*=1 değerlerini ilk çağrı versiyonunda kullanabilirsiniz.

Tarihsel veri istemiyle ilgili detaylı bir örnek için Nesne Bağlama Yöntemleri bölümüne bakınız. Söz konusu örnekte iFractals göstergesinden son 1000 alınması ve son 10 yukarı/aşağı fraktalin çizelge üzerine çizilmesi gösterilmektedir. Benzer bir teknik, eksik veriye sahip olan ve genellikle aşağıda belirtilen stillerle çizilen tüm göstergelerde kullanılabilir:

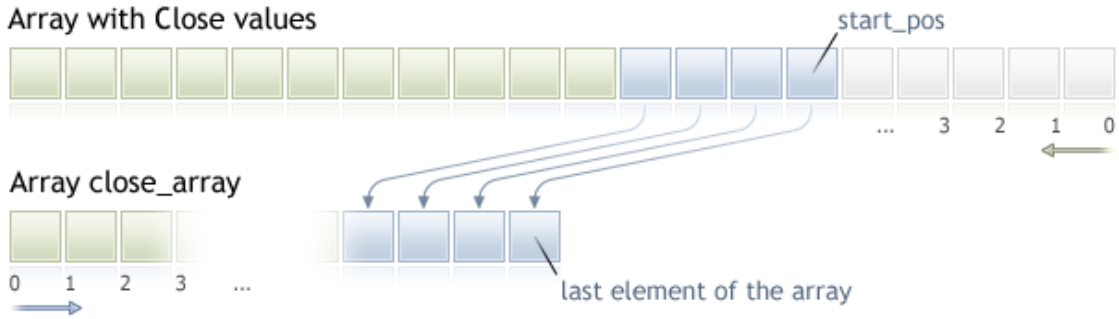
- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

Ayrıca Bakınız

CopyHigh

CopyClose

Seçilen sembol-periyot çifti için, geçmiş kapanış fiyatı verilerini close_array dizisinin içine belirtilen miktarda kopyalar. Elemanların sıralanma şeklinin şimdiden geçmişe doğru olduğu not edilmelidir, yani 0 başlangıç konumu mevcut çubuk anlamına gelecektir.



Bilinmeyen miktardaki bir veriyi kopyalarken, hedef dizi olarak bir dinamik dizinin kullanılması önerilir. Çünkü istenen veri miktarı hedef dizinin büyüklüğünden azsa (veya fazlaysa), fonksiyon, verinin tam olarak uyması için yeni bellek tahsisi yapacaktır.

Eğer kopyalayacağınız verinin tam miktarını biliyorsanız, aşırı bellek tahsisini önlemek amacıyla statik olarak tahsis edilmiş bir tampon kullanmanız daha iyi olacaktır.

Hedef dizisinin erişim özelliğinin ne olduğu önem taşımaz - as_series=true veya as_series=false. Veri, en eski elemanın tahsis edilen fiziksel belleğin başlangıcında konumlanacağı şekilde yerleştirilir. Bu fonksiyonun 3 çağrı şekli bulunmaktadır.

İlk pozisyon ve istenen eleman sayısı ile çağrı

```
int  CopyClose(
   string           symbol_name,       // sembol ismi
   ENUM_TIMEFRAMES  timeframe,         // periyot
   int              start_pos,         // başlangıç konumu
   int              count,             // kopyalanacak veri miktarı
   double           close_array[]      // hedef dizi
   );
```

Başlangıç tarihi ve istenen eleman sayısı ile çağrı

```
int  CopyClose(
   string           symbol_name,       // sembol ismi
   ENUM_TIMEFRAMES  timeframe,         // periyot
   datetime         start_time,        // başlangıç tarihi ve zamanı
   int              count,             // kopyalanacak veri miktarı
   double           close_array[]      // hedef dizi
   );
```

İstene aralığın başlangıç ve bitiş tarihi ile çağrı

```
int  CopyClose(
```

```
   string           symbol_name,       // sembol ismi
   ENUM_TIMEFRAMES  timeframe,         // periyot
   datetime         start_time,        // başlangıç tarihi ve zamanı
   datetime         stop_time,         // bitiş tarihi ve zamanı
   double           close_array[]      // hedef dizi
   );
```

Parametreler

symbol_name

[in] Sembol ismi.

timeframe

[in] Periyot.

start_pos

[in] Kopyalanacak ilk eleman için başlangıç konumu.

count

[in] Kopyalanacak veri miktarı.

start_time

[in] Kopyalanacak ilk eleman için başlangıç zamanı.

stop_time

[in] Kopyalanacak son elemana karşılık gelen çubuğun zamanı.

close_array[]

[out] double tipli dizi.

Dönüş değeri

Kopyalanan veri sayısına veya hata durumunda -1 değerine dönüş yapar.

Not

İstenen veri aralığı sunucuda bulunan mevcut veri aralığının dışındaysa, fonksiyon -1 dönüşü yapar. Eğer istenen veri miktarı TERMINAL_MAXBARS değerinin üstündeyse (çizelgedeki maksimum çubuk sayısı), fonksiyon yine -1 dönüşü yapacaktır.

Bir göstergeden veri istenirken, istenen zaman-serisi henüz kurulmamışsa veya sunucudan yüklenmemişse, fonksiyon hemen -1 dönüşü yapacaktır ve yükleme/kurma işlemi başlatılacaktır.

Bir Uzman Danışmandan veya bir scriptten veri istenirken, istenen veri terminalde yerel olarak bulunmuyorsa, sunucudan yükleme işlemi başlatılacaktır, eğer veri kurulu durumdaysa ama hazır değilse, o zaman serinin kurulumuna başlanacaktır. Varsayılan zaman-aşımı süresi geçildiğinde, fonksiyon hazır olan verinin miktarına dönüş yapacaktır ama geçmiş yüklemesi devam edecektir ve bir sonraki benzer istekte, fonksiyon daha fazla veri dönüşü yapacaktır.

Başlangıç tarihi ve istenen veri miktarını kullanarak bir istek gerçekleştirildiğinde, sadece belirtilen tarihe eşit veya ondan daha önceki verilerin dönüşü yapılacaktır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman belirtilenden az veya belirtilene eşit olacaktır.

Belirtilen tarih aralığındaki bir veri istendiğinde, sadece bu aralıktaki verilere dönüş yapılır. Zaman aralığı ayarlanır ve saniyeye kadar sayılır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman istenilen aralık içindedir.

Bu şekilde, *start_time=Last_Tuesday* ve *stop_time=Last_Friday* aralığındaki bir haftalık verinin kopyalanması isteniyorsa ve mevcut gün cumartesi ise, fonksiyon 0 dönüşü yapacaktır. Çünkü bir haftalık zaman aralığında açılış günü her zaman Pazar günüdür ve bir haftalık çubuk belirtilen aralığa düşmez.

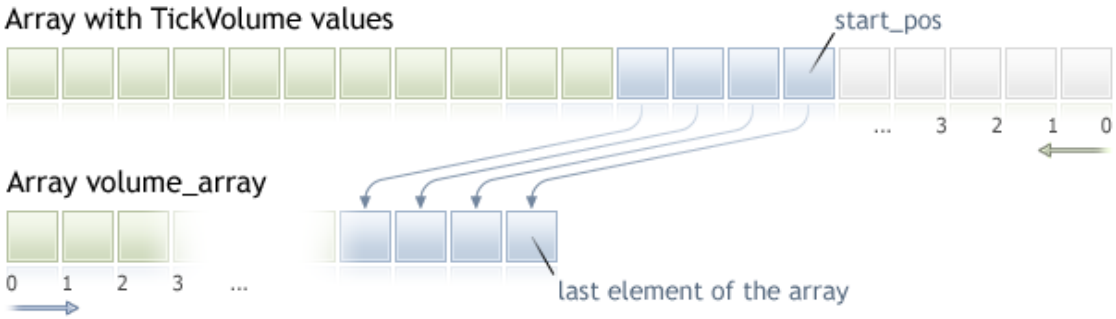
Eğer tamamlanmamış mevcut çubuğun verisine dönüş yaptırmak istiyorsanız, *start_pos*=0 ve *count*=1 değerlerini ilk çağrı versiyonunda kullanabilirsiniz.

Tarihsel verinin istenmesine dair detaylı bir örnek için Nesne Bağlama Yöntemleri bölümüne bakınız. Söz konusu örnekte iFractals göstergesinden son 1000 alınması ve son 10 yukarı/aşağı fraktalin çizelge üzerine çizilmesi gösterilmektedir. Benzer bir teknik, eksik veriye sahip olan ve genellikle aşağıda belirtilen stillerle çizilen tüm göstergelerde kullanılabilir:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

CopyTickVolume

Seçilen sembol-periyot çifti için, geçmiş tik hacmi verilerini belirtilen miktarda volume_array dizisinin içine kopyalar. Elemanların sıralanma şeklinin şimdiden geçmişe doğru olduğu not edilmelidir, yani 0 başlangıç konumu mevcut çubuk anlamına gelecektir.



Bilinmeyen miktardaki bir veriyi kopyalarken, hedef dizi olarak bir dinamik dizinin kullanılması önerilir. Çünkü istenen veri miktarı hedef dizinin büyüklüğünden azsa (veya fazlaysa), fonksiyon, verinin tam olarak uyması için yeni bellek tahsisi yapacaktır.

Eğer kopyalayacağınız verinin tam miktarını biliyorsanız, aşırı bellek tahsisini önlemek amacıyla statik olarak tahsis edilmiş bir tampon kullanmanız daha iyi olacaktır.

Hedef dizisinin erişim özelliğinin ne olduğu önem taşımaz - as_series=true veya as_series=false. Veri, en eski elemanın tahsis edilen fiziksel belleğin başlangıcında konumlanacağı şekilde yerleştirilir. Bu fonksiyonun 3 çağrı şekli bulunmaktadır.

İlk pozisyon ve istenen eleman sayısı ile çağrı

```
int  CopyTickVolume(
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   int              start_pos,        // başlangıç konumu
   int              count,            // kopyalanacak veri miktarı
   long             volume_array[]    // hacim değerleri için hedef dizi
   );
```

Başlangıç tarihi ve istenen eleman sayısı ile çağrı

```
int  CopyTickVolume(
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   datetime         start_time,       // başlangıç tarihi ve zamanı
   int              count,            // kopyalanacak veri miktarı
   long             volume_array[]    // hacim değerleri için hedef dizi
   );
```

İstene aralığın başlangıç ve bitiş tarihi ile çağrı

```
int  CopyTickVolume(
```

```
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   datetime         start_time,       // başlangıç tarihi ve zamanı
   datetime         stop_time,        // bitiş tarihi ve zamanı
   long             volume_array[]    // hacim değerleri için hedef dizi
   );
```

Parametreler

symbol_name

[in] Sembol ismi.

timeframe

[in] Periyot.

start_pos

[in] Kopyalanacak ilk eleman için başlangıç konumu.

count

[in] Kopyalanacak veri miktarı.

start_time

[in] Kopyalanacak ilk eleman için başlangıç zamanı.

stop_time

[in] Kopyalanacak son elemana karşılık gelen çubuğun zamanı.

volume_array[]

[out] long tipli dizi.

Dönüş değeri

Kopyalanan veri sayısına veya hata durumunda -1 değerine dönüş yapar.

Not

İstenen veri aralığı sunucuda bulunan mevcut veri aralığının dışındaysa, fonksiyon -1 dönüşü yapar. Eğer istenen veri miktarı TERMINAL_MAXBARS değerinin üstündeyse (çizelgedeki maksimum çubuk sayısı), fonksiyon yine -1 dönüşü yapacaktır.

Bir göstergeden veri istenirken, istenen zaman-serisi henüz kurulmamışsa veya sunucudan yüklenmemişse, fonksiyon hemen -1 dönüşü yapacaktır ve yükleme/kurma işlemi başlatılacaktır.

Bir Uzman Danışmandan veya bir scriptten veri istenirken, istenen veri terminalde yerel olarak bulunmuyorsa sunucudan yükleme işlemi başlatılacaktır, eğer veri kurulu durumdaysa ama hazır değilse, o zaman serinin kurulumuna başlanacaktır. Varsayılan zaman-aşımı süresi geçildiğinde, fonksiyon hazır olan verinin miktarına dönüş yapacaktır ama geçmiş yüklemesi devam edecektir ve bir sonraki benzer istekte, fonksiyon daha fazla veri dönüşü yapacaktır.

Başlangıç tarihi ve istenen veri miktarını kullanarak bir istek gerçekleştirildiğinde, sadece belirtilen tarihe eşit veya ondan daha önceki verilerin dönüşü yapılacaktır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman belirtilenden az veya belirtilene eşit olacaktır.

Belirtilen tarih aralığındaki bir veri istendiğinde, sadece bu aralıktaki verilere dönüş yapılır. Zaman aralığı ayarlanır ve saniyeye kadar sayılır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman istenilen aralık içindedir.

Bu şekilde, *start_time=Last_Tuesday* ve *stop_time=Last_Friday* aralığındaki bir haftalık verinin kopyalanması isteniyorsa ve mevcut gün cumartesi ise, fonksiyon 0 dönüşü yapacaktır. Çünkü bir haftalık zaman aralığında açılış günü her zaman Pazar günüdür ve bir haftalık çubuk belirtilen aralığa düşmez.

Eğer tamamlanmamış mevcut çubuğun verisine dönüş yaptırmak istiyorsanız, *start_pos*=0 ve *count*=1 değerlerini ilk çağrı versiyonunda kullanabilirsiniz.

Örnek:

```
#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
//--- giriş parametreleri
input int      bars=3000;
//--- gösterge tamponları
double         TickVolumeBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,TickVolumeBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[],
```

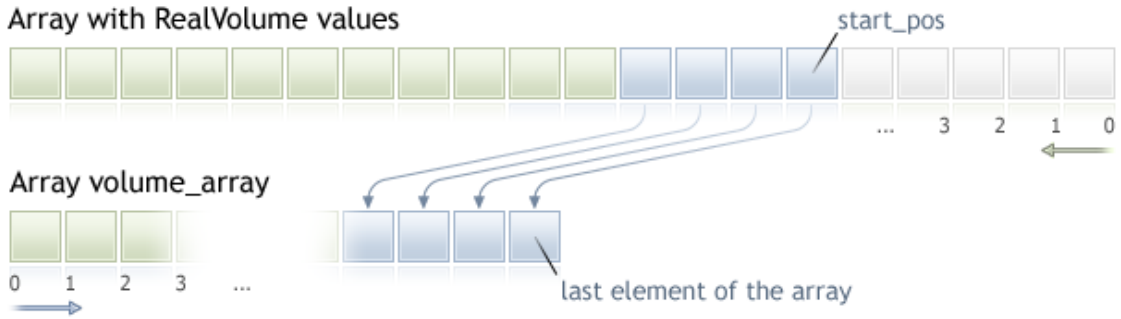
```
                  const int &spread[])
  {
//---
   if(prev_calculated==0)
     {
      long timeseries[];
      ArraySetAsSeries(timeseries,true);
      int prices=CopyTickVolume(Symbol(),0,0,bars,timeseries);
      for(int i=0;i<rates_total-prices;i++) TickVolumeBuffer[i]=0.0;
      for(int i=0;i<prices;i++) TickVolumeBuffer[rates_total-1-i]=timeseries[prices-1-
      Print("Şu sayıda Tik Hacmi verisi aldık: "+prices);
     }
   else
     {
      long timeseries[];
      int prices=CopyTickVolume(Symbol(),0,0,1,timeseries);
      TickVolumeBuffer[rates_total-1]=timeseries[0];
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Tarihsel verinin istenmesine dair detaylı bir örnek için Nesne Bağlama Yöntemleri bölümüne bakınız. Söz konusu örnekte iFractals göstergesinden son 1000 alınması ve son 10 yukarı/aşağı fraktalin çizelge üzerine çizilmesi gösterilmektedir. Benzer bir teknik, eksik veriye sahip olan ve genellikle aşağıda belirtilen stillerle çizilen tüm göstergelerde kullanılabilir:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

CopyRealVolume

Seçilen sembol-periyot çifti için, geçmiş alım-satım hacmi verilerini belirtilen miktarda volume_array dizisinin içine kopyalar. Elemanların sıralanma şeklinin şimdiden geçmişe doğru olduğu not edilmelidir, yani 0 başlangıç konumu mevcut çubuk anlamına gelecektir.



Bilinmeyen miktardaki bir veriyi kopyalarken, hedef dizi olarak bir dinamik dizinin kullanılması önerilir. Çünkü istenen veri miktarı hedef dizinin büyüklüğünden azsa (veya fazlaysa), fonksiyon, verinin tam olarak uyması için yeni bellek tahsisi yapacaktır.

Eğer kopyalayacağınız verinin tam miktarını biliyorsanız, aşırı bellek tahsisini önlemek amacıyla statik olarak tahsis edilmiş bir tampon kullanmanız daha iyi olacaktır.

Hedef dizisinin erişim özelliğinin ne olduğu önem taşımaz - as_series=true veya as_series=false. Veri, en eski elemanın tahsis edilen fiziksel belleğin başlangıcında konumlanacağı şekilde yerleştirilir. Bu fonksiyonun 3 çağrı şekli bulunmaktadır.

İlk pozisyon ve istenen eleman sayısı ile çağrı

```
int  CopyRealVolume(
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   int              start_pos,        // başlangıç konumu
   int              count,            // kopyalanacak veri miktarı
   long             volume_array[]    // hacim değerleri için hedef dizi
   );
```

Başlangıç tarihi ve istenen eleman sayısı ile çağrı

```
int  CopyRealVolume(
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   datetime         start_time,       // başlangıç tarihi ve zamanı
   int              count,            // kopyalanacak veri miktarı
   long             volume_array[]    // hacim değerleri için hedef dizi
   );
```

İstene aralığın başlangıç ve bitiş tarihi ile çağrı

```
int  CopyRealVolume(
```

```
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   datetime         start_time,       // başlangıç tarihi ve zamanı
   datetime         stop_time,        // bitiş tarihi ve zamanı
   long             volume_array[]    // hacim değerleri için hedef dizi
   );
```

Parametreler

symbol_name

[in] Sembol ismi.

timeframe

[in] Periyot.

start_pos

[in] Kopyalanacak ilk eleman için başlangıç konumu.

count

[in] Kopyalanacak veri miktarı.

start_time

[in] Kopyalanacak ilk eleman için başlangıç zamanı.

stop_time

[in] Kopyalanacak son elemana karşılık gelen çubuğun zamanı.

volume_array[]

[out] long tipli dizi.

Dönüş değeri

Kopyalanan veri sayısına veya hata durumunda -1 değerine dönüş yapar.

Not

İstenen veri aralığı sunucuda bulunan mevcut veri aralığının dışındaysa, fonksiyon -1 dönüşü yapar. Eğer istenen veri miktarı TERMINAL_MAXBARS değerinin üstündeyse (çizelgedeki maksimum çubuk sayısı), fonksiyon yine -1 dönüşü yapacaktır.

Bir göstergeden veri istenirken, istenen zaman-serisi henüz kurulmamışsa veya sunucudan yüklenmemişse, fonksiyon hemen -1 dönüşü yapacaktır ve yükleme/kurma işlemi başlatılacaktır.

Bir Uzman Danışmandan veya bir scriptten veri istenirken, istenen veri terminalde yerel olarak bulunmuyorsa sunucudan yükleme işlemi başlatılacaktır, eğer veri kurulu durumdaysa ama hazır değilse, o zaman serinin kurulumuna başlanacaktır. Varsayılan zaman-aşımı süresi geçildiğinde, fonksiyon hazır olan verinin miktarına dönüş yapacaktır ama geçmiş yüklemesi devam edecektir ve bir sonraki benzer istekte, fonksiyon daha fazla veri dönüşü yapacaktır.

Başlangıç tarihi ve istenen veri miktarını kullanarak bir istek gerçekleştirildiğinde, sadece belirtilen tarihe eşit veya ondan daha önceki verilerin dönüşü yapılacaktır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman belirtilenden az veya belirtilene eşit olacaktır.

Belirtilen tarih aralığındaki bir veri istendiğinde, sadece bu aralıktaki verilere dönüş yapılır. Zaman aralığı ayarlanır ve saniyeye kadar sayılır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman istenilen aralık içindedir.

Bu şekilde, *start_time=Last_Tuesday* ve *stop_time=Last_Friday* aralığındaki bir haftalık verinin kopyalanması isteniyorsa ve mevcut gün cumartesi ise, fonksiyon 0 dönüşü yapacaktır. Çünkü bir haftalık zaman aralığında açılış günü her zaman Pazar günüdür ve bir haftalık çubuk belirtilen aralığa düşmez.

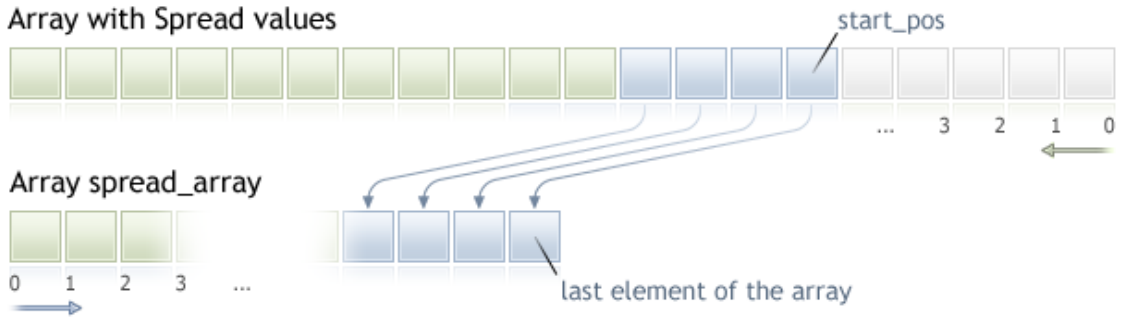
Eğer tamamlanmamış mevcut çubuğun verisine dönüş yaptırmak istiyorsanız, *start_pos*=0 ve *count*=1 değerlerini ilk çağrı versiyonunda kullanabilirsiniz.

Tarihsel verinin istenmesine dair detaylı bir örnek için Nesne Bağlama Yöntemleri bölümüne bakınız. Söz konusu örnekte iFractals göstergesinden son 1000 alınması ve son 10 yukarı/aşağı fraktalin çizelge üzerine çizilmesi gösterilmektedir. Benzer bir teknik, eksik veriye sahip olan ve genellikle aşağıda belirtilen stillerle çizilen tüm göstergelerde kullanılabilir:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

CopySpread

Seçilen sembol-periyot çifti için, geçmiş makas verilerini spread_array dizisine belirtilen miktarda kopyalar. Elemanların sıralanma şeklinin şimdiden geçmişe doğru olduğu not edilmelidir, yani 0 başlangıç konumu mevcut çubuk anlamına gelecektir.



Bilinmeyen miktardaki bir veriyi kopyalarken, hedef dizi olarak bir dinamik dizinin kullanılması önerilir. Çünkü istenen veri miktarı hedef dizinin büyüklüğünden azsa (veya fazlaysa), fonksiyon, verinin tam olarak uyması için yeni bellek tahsisi yapacaktır.

Eğer kopyalayacağınız verinin tam miktarını biliyorsanız, aşırı bellek tahsisini önlemek amacıyla statik olarak tahsis edilmiş bir tampon kullanmanız daha iyi olacaktır.

Hedef dizisinin erişim özelliğinin ne olduğu önem taşımaz - as_series=true veya as_series=false. Veri, en eski elemanın tahsis edilen fiziksel belleğin başlangıcında konumlanacağı şekilde yerleştirilir. Bu fonksiyonun 3 çağrı şekli bulunmaktadır.

İlk pozisyon ve istenen eleman sayısı ile çağrı

```
int  CopySpread(
   string            symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES   timeframe,        // periyot
   int               start_pos,        // başlangıç konumu
   int               count,            // kopyalanacak veri miktarı
   int               spread_array[]    // makas değerleri için hedef dizi
   );
```

Başlangıç tarihi ve istenen eleman sayısı ile çağrı

```
int  CopySpread(
   string            symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES   timeframe,        // periyot
   datetime          start_time,       // başlangıç tarihi ve zamanı
   int               count,            // kopyalanacak veri miktarı
   int               spread_array[]    // makas değerleri için hedef dizi
   );
```

İstene aralığın başlangıç ve bitiş tarihi ile çağrı

```
int  CopySpread(
```

```
   string           symbol_name,      // sembol ismi
   ENUM_TIMEFRAMES  timeframe,        // periyot
   datetime         start_time,       // başlangıç tarihi ve zamanı
   datetime         stop_time,        // bitiş tarihi ve zamanı
   int              spread_array[]    // makas değerleri için hedef dizi
   );
```

Parametreler

symbol_name

[in] Sembol ismi.

timeframe

[in] Periyot.

start_pos

[in] Kopyalanacak ilk eleman için başlangıç konumu.

count

[in] Kopyalanacak veri miktarı.

start_time

[in] Kopyalanacak ilk eleman için başlangıç zamanı.

stop_time

[in] Kopyalanacak son elemana karşılık gelen çubuğun zamanı.

spread_array[]

[out] int tipli dizi.

Dönüş değeri

Kopyalanan veri sayısına veya hata durumunda -1 değerine dönüş yapar.

Not

İstenen veri aralığı sunucuda bulunan mevcut veri aralığının dışındaysa, fonksiyon -1 dönüşü yapar. Eğer istenen veri miktarı TERMINAL_MAXBARS değerinin üstündeyse (çizelgedeki maksimum çubuk sayısı), fonksiyon yine -1 dönüşü yapacaktır.

Bir göstergeden veri istenirken, istenen zaman-serisi henüz kurulmamışsa veya sunucudan yüklenmemişse, fonksiyon hemen -1 dönüşü yapacaktır ve yükleme/kurma işlemi başlatılacaktır.

Bir Uzman Danışmandan veya bir scriptten veri istenirken, istenen veri terminalde yerel olarak bulunmuyorsa sunucudan yükleme işlemi başlatılacaktır, eğer veri kurulu durumdaysa ama hazır değilse, o zaman serinin kurulumuna başlanacaktır. Varsayılan zaman-aşımı süresi geçildiğinde, fonksiyon hazır olan verinin miktarına dönüş yapacaktır ama geçmiş yüklemesi devam edecektir ve bir sonraki benzer istekte, fonksiyon daha fazla veri dönüşü yapacaktır.

Başlangıç tarihi ve istenen veri miktarını kullanarak bir istek gerçekleştirildiğinde, sadece belirtilen tarihe eşit veya ondan daha önceki verilerin dönüşü yapılacaktır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman belirtilenden az veya belirtilene eşit olacaktır.

Belirtilen tarih aralığındaki bir veri istendiğinde, sadece bu aralıktaki verilere dönüş yapılır. Zaman aralığı ayarlanır ve saniyeye kadar sayılır. Yani, değerine dönüş yapılacak herhangi bir çubuğun açılış zamanı (hacim, makas, gösterge tamponuna dair bir değer, OHLC fiyatları ve Time açılış zamanı), her zaman istenilen aralık içindedir.

Bu şekilde, *start_time=Last_Tuesday* ve *stop_time=Last_Friday* aralığındaki bir haftalık verinin kopyalanması isteniyorsa ve mevcut gün cumartesi ise, fonksiyon 0 dönüşü yapacaktır. Çünkü bir haftalık zaman aralığında açılış günü her zaman Pazar günüdür ve bir haftalık çubuk belirtilen aralığa düşmez.

Eğer tamamlanmamış mevcut çubuğun verisine dönüş yaptırmak istiyorsanız, *start_pos*=0 ve *count*=1 değerlerini ilk çağrı versiyonunda kullanabilirsiniz.

Örnek:

```
#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
//--- giriş parametreleri
input int      bars=3000;
//--- gösterge tamponları
double         SpreadBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,SpreadBuffer,INDICATOR_DATA);
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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("Şu kadar Makas verisi aldık: ",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();
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Tarihsel verinin istenmesine dair detaylı bir örnek için Nesne Bağlama Yöntemleri bölümüne bakınız. Söz konusu örnekte iFractals göstergesinden son 1000 alınması ve son 10 yukarı/aşağı fraktalin çizelge üzerine çizilmesi gösterilmektedir. Benzer bir teknik, eksik veriye sahip olan ve genellikle aşağıda belirtilen stillerle çizilen tüm göstergelerde kullanılabilir:

- DRAW_SECTION,
- DRAW_ARROW,
- DRAW_ZIGZAG,
- DRAW_COLOR_SECTION,
- DRAW_COLOR_ARROW,
- DRAW_COLOR_ZIGZAG.

CopyTicks

MqlTick biçimindeki tik verilerini ticks_array dizisine aktarır. Tik verileri geçmişten şimdiye doğru indislenir, yani 0 indisi dizideki en eski tik verisini gösterir. Tik analizi için, tik üzerinde tam olarak neyin değiştiğini gösteren *flags* alanını kullanın.

```
int  CopyTicks(
   string           symbol_name,           // Sembol ismi
   MqlTick&         ticks_array[],         // Tik verilerinin yükleneceği dizi
   uint             flags=COPY_TICKS_ALL,  // İstenen tik verilerinin tipini tanımlaya
   ulong            from=0,                // İstenen tik verilerinin başlangıç tarihi
   uint             count=0                // Alınmak istenen tik verilerinin sayısı
   );
```

Parametreler

symbol_name

[in] Sembol.

ticks_array

[out] Tik verilerinin alınması için MqlTick tipinde bir dizi.

bayraklar

[in] İstenen tik verilerinin tipini tanımlayacak bayrak. COPY_TICKS_INFO - Satış ve/veya Alış değişimlerini içeren tik verileri, COPY_TICKS_TRADE - Son fiyat ve hacim değişimlerini içeren tik verileri, COPY_TICKS_ALL - tüm tikler. Tüm istek tiplerinde, önceki tik değerleri MqlTick yapısının kalan alanlarına eklenir.

from

[in] İstenen tik verilerinin başlangıç tarihi. 1970.01.01 tarihinden itibaren milisaniye cinsinden. *from*=0 ise, *count* ile belirtilen miktar kadar tik verisi alınır.

count

[in] İstenen verilerin sayısı. 'from' ve 'count' parametreleri belirtilmemişse, tüm mevcut tik verileri (2000 'den fazla olmamak kaydıyla) ticks_array[] dizisine yazılır.

Dönüş değeri

Kopyalanan tiklerin sayısı veya hata durumunda -1.

Notlar

CopyTicks() fonksiyonu tik verilerinin alınmasını ve analiz edilmesini sağlar. CopyTicks() fonksiyonunun ilk çağrısı ile sembolün sabit disk üzerindeki tik veri-tabanının eşitlenme işlemi başlatılır. İstenen verilerin bir kısmı yerel veri-tabanında mevcut değilse, eksik veriler otomatik olarak alım-satım sunucusundan indirilir. CopyTicks() içerisinde ***from*** ile belirtilem tarihten şu ana kadar olan tüm tikler eşitlenir. Daha sonra gelen tik verilerinin hepsi otomatik olarak veri-tabanına eklenir. Bu sayede veriler güncel tutulur.

from ve *count* parametreleri belirtilmemişse, tüm mevcut tik verileri (2000 'den fazla olmamak kaydıyla) *ticks_array[]* dizisine yazılır. *flags* parametresi istenen tik verilerinin tipinin belirtilmesini sağlar.

COPY_TICKS_INFO - Satış ve/veya Alış değişimlerini içeren tik verilerine dönüş yapar. Ayrıca diğer alanlardaki veriler de eklenir. Örneğin, sadece Satış fiyatı değişmiş olsa bile *Alış* ve *hacim* alanları da bilinen en son değerler ile doldurulur. Hangi verinin değiştiğini öğrenmek için *flags* alanını kontrol edin (bu alanda TICK_FLAG_BID ve/veya TICK_FLAG_ASK değerleri yer alır). Tik verisi içinde Alış ve Satış fiyatları aynı anda sıfır değeri almışsa ve flags alanı verinin değiştiğini gösteriyorsa (*flags*=TICK_FLAG_BID|TICK_FLAG_ASK), bu durum Piyasa Derinliğinin boş olduğunu ifade eder. Diğer bir deyişle herhangi bir alış veya satış emri yoktur.

COPY_TICKS_TRADE - Son fiyat ve hacim değişimlerini içeren tik verilerine dönüş yapar. Ayrıca diğer alanlardaki veriler de eklenir. Alış ve Şatış alanları da bilinen en son değerler ile doldurulur. Hangi verinin değiştiğini öğrenmek için *flags* alanını kontrol edin (bu alanda TICK_FLAG_LAST ve TICK_FLAG_VOLUME değerleri yer alır).

COPY_TICKS_ALL - herhangi bir değişim durumunda tüm tik verilerine dönüş yapılır. Değişmeyen alanlar bilinen en son değerler ile doldurulur

COPY_TICKS_ALL bayrağı ile yapılan CopyTicks() çağrısı istenen aralıktaki tüm tik verilerine hemen dönüş yaparken, diğer tipteki isteklerin işlenmesi daha fazla zaman alır. Bu yüzden diğer istek tiplerinde hız avantajı sağlanamaz.

COPY_TICKS_INFO veya COPY_TICKS_TRADE ile istenen tik verileri, değişim anındaki tüm fiyat bilgilerini içerirler (*satış, alış, son* fiyatları ve *hacim*). Bu özellik veri değişim anındaki işlem durumunu kolayca incelemek için tasarlanmıştır. Bu sayede tik geçmişi üzerindeki diğer alanları derinlemesine incelemeye gerek operati olmaz.

CopyTicks() fonksiyonu gösterge içinde çağrıldığında sonuca dönüş yapar: sembolün var olan tüm tik verilerine dönüş yapılır ve yeterli tik verisi yoksa tik veri-tabanının eşitleme işlemini başlatır. Belli bir sembol ile açılan tüm göstergeler aynı iş parçacığı üzerinde çalışır. Bu sayede eşitleme işleminin bitmesi beklenmez. Eşitlemenin ardından CopyTicks() fonksiyonu bir sonraki çağrıda tüm tik verilerine dönüş yapar. Göstergelerde OnCalculate() fonksiyonu yeni tik verisi alındıktan sonra çağrılır.

CopyTicks() fonksiyonu Uzmanlarda ve betiklerde 45 saniye süreyle sonuç bekleyebilir: Uzman Danışmanlar ve betikler -göstergelerden farklı olarak- ayrı iş parçacıkları üzerinde çalışırlar bu nedenle eşitleme işlemi için 45 saniye kadar bekleyebilirler. Bu süre içinde istenen veri miktarı eşitlenemezse, CopyTicks() sürenin sonunda elde edilen verilere dönüş yapar ve eşitleme işlemine devam eder. Uzman Danışmanlarda OnTick() fonksiyonu bir tik işleyici değildir, sadece Uzman Danışmanı piyasadaki değişimler hakkında bilgilendirir. Bu tek bir değişim olmayabilir: terminal aynı anda birkaç tik verisi oluşturabilir ama OnTick() fonksiyonu piyasa durumu hakkında Uzmanı bilgilendirmek için sadece bir kez çağrılır.

Veri dönüş oranı: terminal her bir enstrümanın son 4096 tik verisini hızlı erişim deposunda tutar (Piyasa Derinliğine sahip olan semboller için 65536 tik). Mevcut işlem seansı için istenen tiklerin sayısı depoda bulunandan fazlaysa, CopyTicks() fonksiyonu terminal belleğinde depolanan tik verilerini de alır. Bu isteklerin uygulanması daha fazla zaman alır. En yavaş işlenenler istekler başka günlerin tik verilerinin istenmesi ile ortaya çıkar.

Örnek:

```
#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
```

```
//--- Show the time of the last tick in the array
   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)
*/
```

Ayrıca bakınız

SymbolInfoTick, Cari fiyatların Yapısı, OnTick()

CopyTicksRange

Belirtilen aralıktaki tik fiyatlarını MqlTick biçiminde ticks_array dizisine depolar. Numaralandırma geçmişten bugüne doğru yapılır, yani 0 indisli veri dizideki en eski tik fiyatıdır. Tik analizi için, tam olarak neyin değiştiğini gösteren *flags* alanını kontrol edin.

```
int  CopyTicksRange(
   const string      symbol_name,           // sembol ismi
   MqlTick&          ticks_array[],         // hedef dizi
   uint              flags=COPY_TICKS_ALL,  // depolanacak tiklerin tipini belirten bay
   ulong             from_msc=0,            // alınacak tik verileri için başlangıç tar
   ulong             to_msc=0               // alınacak tik verileri için son tarih
   );
```

Parametreler

symbol_name

[in] Sembol.

ticks_array

[out] MqlTick biçimli statik veta dinamik dizi. Statik dizi belirtilen aralıktaki tüm tik verilerini alacak kadar geniş değilse, dizinin kapasitesine göre maksimum tik verisi depolanır. Bu durumda fonksiyon şu hatayı verir: ERR_HISTORY_SMALL_BUFFER (4407).

bayraklar

[in] İstenen tik verilerinin tipini tanımlayacak bayrak. COPY_TICKS_INFO - Satış ve/veya Alış değişimlerini içeren tik verileri, COPY_TICKS_TRADE - Son fiyat ve hacim değişimlerini içeren tik verileri, COPY_TICKS_ALL - tüm tikler. Tüm istek tiplerinde, önceki tik değerleri MqlTick yapısının kalan alanlarına eklenir.

from_msc

[in] İstenen tik verilerinin başlangıç tarihi. 1970.01.01 tarihinden itibaren milisaniye cinsinden. *from_msc* parametresi belirtilmemişse mevcut olan en eski tik verisinden başlanır. Tik zamanı from_msc değerinden büyük olan veriler alınır.

to_msc

[in] alınacak tik verileri için son tarih. 01.01.1970 tarihinden itibaren milisaniye cinsinden. Tik zamanı to_msc değerinden küçük olan veriler alınır. *to_msc* parametresi belirtilmemişse mevcut olan en son tarihli tik verisine kadar tüm veriler alınır.

Dönüş Değeri

Kopyalanan tiklerin sayısı veya hata durumunda -1. GetLastError() çağrısı şu hatalara dönüş yapabilir:

- ERR_HISTORY_TIMEOUT - tik senkronizasyonu için tanımlanan süre aşıldı, fonksiyon sahip olunan verilerle çağrıldı.
- ERR_HISTORY_SMALL_BUFFER - statik tampon çok küçük. Sadece dizi boyutu kadar veri alınabilir.
- ERR_NOT_ENOUGH_MEMORY - belirtilen aralıktaki verileri depolamak için yeterli bellek yok. Tik dizisi için yeterli bellek tahsis edilemedi.

Not

CopyTicksRange() fonksiyonu belli bir tarih aralığındaki tik verilerini almak için kullanılır, örneğin geçmiş üzerindeki bir günün verileri. CopyTicks() fonksiyonu ise sadec bir başlangıç tarihi belirtebilebilir, örneğin – örneğin ayın başından şu ana kadar olan tik verilerinin alınması.

Ayrıca bakınız

SymbolInfoTick, Mevcut Fiyatlar için Yapı, OnTick, CopyTicks

iBars

Tarihte mevcut karşılık gelen sembol ve dönemin çubuk sayısını döndürür.

```
int  iBars(
   const string           symbol,          // Sembol
   ENUM_TIMEFRAMES        timeframe        // Periyot
   );
```

Parametreler

symbol

[in] Finansal enstrümanın sembol ismi. NULL şimdiki sembol anlamına gelir.

timeframe

[in] Periyot. ENUM_TIMEFRAMES sayısının değerlerinden biri olabilir. 0 şimdiki grafik periyodu anlamına gelir.

Return Value

Tarihçede kullanılabilen ancak platform ayarlarında "Grafikteki maksimum çubuklar" parametresi tarafından izin verilen en fazla karşılık gelen sembol ve döneme ait çubuk sayısı.

Örnek:

```
  Print("Bar count on the 'EURUSD,H1' is ",iBars("EURUSD",PERIOD_H1));
```

Ayrıca bakınız

Bars

iBarShift

Zamana göre bar arama. İşlev, belirtilen zamana karşılık gelen çubuğun dizinini döndürür.

```
int  iBarShift(
   const string        symbol,          // Sembol
   ENUM_TIMEFRAMES     timeframe,       // Periyot
   datetime            time,            // Zaman
   bool                exact=false      // Mod
   );
```

Parametreler

symbol

[in] Finansal enstrümanın sembol ismi. NULL şimdiki sembol anlamına gelir.

timeframe

[in] Periyot. ENUM_TIMEFRAMES sayısının değerlerinden biri olabilir. PERIOD_CURRENT şimdiki grafik periyodu anlamına gelir.

time

[in] Arama için zaman değeri.

exact=false

[in] Belirtilen zamana sahip çubuğun bulunmaması durumunda bir dönüş değeri. Eğer *exact=false*, *iBarShift* en yakın barın indeksini döndürür, Açık kalma süresi belirtilen süreden daha azsa (*time_open<time*). Böyle bir çubuk bulunmazsa (belirtilen zamandan önce geçmiş mevcut değil), işlev -1 döndürür. Eğer *exact=true*, iBarShift en yakın çubuğu aramaz ancak hemen -1 döndürür.

Return Value

Belirtilen zamana karşılık gelen çubuğun indeksi. Belirtilen zamana karşılık gelen çubuk bulunmazsa (tarihte bir boşluk vardır), işlev -1 değerini veya en yakın çubuğun indisini (*'exact'* değerine bağlı olarak döndürür.

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Tarih Pazardır.
   datetime time=D'2002.04.25 12:00';
   string symbol="GBPUSD";
   ENUM_TIMEFRAMES tf=PERIOD_H1;
   bool exact=false;
//--- Belirtilen zamanda herhangi bir çubuk yoksa, iBarShift en yakın çubuğun dizinini
   int bar_index=iBarShift(symbol,tf,time,exact);
//--- IBarShift() çağrısından sonra hata kodunu kontrol edin
   int error=GetLastError();
   if(error!=0)
```

```
      {
       PrintFormat("iBarShift(): GetLastError=%d - The requested date %s "+
                   "for %s %s is not found in the available history",
                   error,TimeToString(time),symbol,EnumToString(tf));
       return;
      }
//--- IBarShift() işlevi başarılı bir şekilde yürütüldü, exact=false için bir sonuç dö
   PrintFormat("1. %s %s %s(%s): bar index is %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time),
               DayOfWeek(time),bar_index,string(exact));
   datetime bar_time=iTime(symbol,tf,bar_index);
   PrintFormat("Time of bar #%d is %s (%s)",
               bar_index,TimeToString(bar_time),DayOfWeek(bar_time));
//--- Çubuğun endeksini belirtilen süre ile isteyin; bar yoksa -1 iade edilir
   exact=true;
   bar_index=iBarShift(symbol,tf,time,exact);
//--- IBarShift() işlevi başarıyla yürütüldü, exact=true için bir sonuç döner
   PrintFormat("2. %s %s %s (%s):bar index is %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time)
               ,DayOfWeek(time),bar_index,string(exact));
  }
//+------------------------------------------------------------------+
//| Haftanın gününün ismini döndürme                                 |
//+------------------------------------------------------------------+
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;
  }
```

```
//+------------------------------------------------------------------+
/* Yürütme sonucu
   1. GBPUSD PERIOD_H1 2018.06.10 12:00(SUNDAY): bar indeks 64 tür(exact=false)
   Time of bar #64 is 2018.06.08 23:00 (FRIDAY)
   2. GBPUSD PERIOD_H1 2018.06.10 12:00 (SUNDAY):bar indeks -1 (exact=true)
*/
```

iClose

İlgili grafikteki çubuğun Kapanış fiyatını ('shift' parametresiyle gösterilir) döndürür.

```
double  iClose(
   const string         symbol,          // Sembol
   ENUM_TIMEFRAMES      timeframe,       // Periyot
   int                  shift            // Shift
   );
```

Parametreler

symbol

[in] Finansal enstrümanın sembol ismi. NULL şimdiki sembol anlamına gelir.

timeframe

[in] Periyot. ENUM_TIMEFRAMES sayısının değerlerinden biri olabilir. 0 şimdiki grafik periyodu anlamına gelir.

shift

[in] Zaman çizelgelerinden alınan değerin indeksi (mevcut çubuğa göre belirlenen sayıda çubuk ile geriye doğru kaydırma).

Return Value

İlgili grafikteki çubuğun ('shift' parametresi ile gösterilir) kapanış fiyatı veya bir hata durumunda 0. Hata detayları için,GetLastError() fonksiyonunu çağırın.

Not

Fonksiyon her zaman gerçek verileri döndürür. Bu amaçla, her bir arama sırasında belirtilen sembol/periyot için zaman çizelgesine bir istek yapar. Bu, ilk işlev çağrısı sırasında hazır bir veri yoksa, sonucu hazırlamak için biraz zaman alınabileceği anlamına gelir.

İşlev önceki arama sonuçlarını saklamaz ve hızlı değer döndürme için yerel önbellek yoktur.

Example:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Fonksiyon-olay işleyicisi "tik"                                  |
//+------------------------------------------------------------------+
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"
            );
  }
```

Ayrıca bakınız

CopyClose, CopyRates

iHigh

İlgili grafikteki çubuğun Yüksek fiyatını ('shift' parametresiyle gösterilir) döndürür.

```
double  iHigh(
   const string         symbol,          // Sembol
   ENUM_TIMEFRAMES      timeframe,       // Periyot
   int                  shift            // Shift
   );
```

Parametreler

symbol

[in] Finansal enstrümanın sembol ismi. NULL şimdiki sembol anlamına gelir.

timeframe

[in] Periyot. ENUM_TIMEFRAMES sayısının değerlerinden biri olabilir. 0 şimdiki grafik periyodu anlamına gelir.

shift

[in] Zaman çizelgelerinden alınan değerin indeksi (mevcut çubuğa göre belirlenen sayıda çubuk ile geriye doğru kaydırma).

Return Value

İlgili grafikteki çubuğun ('shift' parametresi ile gösterilir) yüksek fiyatı veya bir hata durumunda 0. Hata detayları içinGetLastError() fonksiyonunu çağırın.

Not

Fonksiyon her zaman gerçek değerleri döndürür. Bu amaçla, her bir arama sırasında belirtilen sembol/periyot için zaman çizelgesine bir istek yapar. Bu, ilk işlev çağrısı sırasında hazır bir veri yoksa, sonucu hazırlamak için biraz zaman alınabileceği anlamına gelir.

İşlev önceki arama sonuçlarını saklamaz ve hızlı değer döndürme için yerel önbellek yoktur.

Örnek:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Fonksiyon-olay işleyicisi "tik"                                  |
//+------------------------------------------------------------------+
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"
            );
  }
```

Ayrıca bakınız

CopyHigh, CopyRates

iHighest

İlgili grafikte bulunan en yüksek değerin indeksini döndürür (geçerli çubuğa göre).

```
int  iHighest(
   const string        symbol,              // Sembol
   ENUM_TIMEFRAMES     timeframe,           // Periyot
   ENUM_SERIESMODE     type,                // Zaman serileri belirleyicisi
   int                 count=WHOLE_ARRAY,   // Elementlerin sayısı
   int                 start=0              // İndeks
  );
```

Parametreler

symbol

[in] Aramanın yapılacağı sembol. NULL şimdiki sembol anlamına gelir.

timeframe

[in] Periyot. ENUM_TIMEFRAMES sayısının değerlerinden biri olabilir. 0 şimdiki grafik periyodu anlamına gelir.

type

[in] Aramanın gerçekleştirileceği zaman dizisinin tanımlayıcısı. ENUM_SERIESMODE değerinden herhangi bir değere eşit olabilir.

count=WHOLE_ARRAY

[in] Zaman aralığındaki eleman sayısı (mevcut çubuktan indeks artış yönüne doğru).

start=0

[in] En yüksek değerin aranacağı başlangıç çubuğunun endeksi (mevcut çubuğa göre kaydırma) başlar. Negatif değerler yok sayılır ve sıfır değeri ile değiştirilir.

Return Value

İlgili grafikte bulunan en yüksek değerin indeksi (mevcut çubuğa göre) veya bir hata durumunda -1. Hata detayları için,GetLastError() fonksiyonunu çağırın.

Örnek:

```
   double val;
//--- 20 ardışık çubuk arasında en yüksek Kapanış değerinin hesaplanması
//--- Şu andaki zaman dilimine göre endeks 4'ten endeks 23'e kadar.
   int val_index=iHighest(NULL,0,MODE_CLOSE,20,4);
   if(val_index!=-1)
      val=High[val_index];
   else
      PrintFormat("iHighest() call error. Error code=%d",GetLastError());
```

iLow

İlgili grafikteki çubuğun Düşük fiyatını ('shift' parametresiyle gösterilir) döndürür.

```
double  iLow(
   const string         symbol,          // Sembol
   ENUM_TIMEFRAMES      timeframe,       // Periyot
   int                  shift            // Shift
   );
```

Parametreler

symbol

[in] Finansal enstrümanın sembol ismi. NULL şimdiki sembol anlamına gelir.

timeframe

[in] Periyot. ENUM_TIMEFRAMES sayısının değerlerinden biri olabilir. 0 şimdiki grafik periyodu anlamına gelir.

shift

[in] Zaman çizelgelerinden alınan değerin indeksi (mevcut çubuğa göre belirlenen sayıda çubuk ile geriye doğru kaydırma).

Return Value

Çubuğun düşük fiyatı ('shift' parametresi ile gösterilir) ilgili grafikte ya da bir hata durumunda 0. Hata detayları için,GetLastError() fonksiyonunu çağırın.

Not

Fonksiyon her zaman gerçek verileri döndürür. Bu amaçla, her bir arama sırasında belirtilen sembol/periyot için zaman çizelgesine bir istek yapar. Bu, ilk işlev çağrısı sırasında hazır bir veri yoksa, sonucu hazırlamak için biraz zaman alınabileceği anlamına gelir.

İşlev önceki arama sonuçlarını saklamaz ve hızlı değer döndürme için yerel önbellek yoktur.

Örnek:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Fonksiyon-olay işleyicisi "tik"                                  |
//+------------------------------------------------------------------+
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"
            );
  }
```

Ayrıca bakınız

CopyLow, CopyRates

iLowest

İlgili grafikte bulunan en küçük değerin indeksini döndürür.

```
int  iLowest(
   const string        symbol,              // Sembol
   ENUM_TIMEFRAMES     timeframe,           // Periyot
   ENUM_SERIESMODE     type,                // Zamanserileri belirleyicisi
   int                 count=WHOLE_ARRAY,   // Elementlerin sayısı
   int                 start=0              // İndeks
  );
```

Parametreler

symbol

[in] Aramanın yapılacağı sembol. NULL şimdiki sembol anlamına gelir.

timeframe

[in] Periyot. ENUM_TIMEFRAMES sayısının değerlerinden biri olabilir. 0 şimdiki grafik periyodu anlamına gelir.

type

[in] Aramanın gerçekleştirileceği zaman dizisinin tanımlayıcısı. ENUM_SERIESMODE değerinden herhangi bir değere eşit olabilir.

count=WHOLE_ARRAY

[in] Zaman aralığındaki eleman sayısı (mevcut çubuktan indeks artış yönüne doğru), aramanın yapılması gereken.

start=0

[in] En düşük değerin aranacağı başlangıç çubuğunun indeksi (mevcut çubuğa göre kaydırma) başlar. Negatif değerler yok sayılır ve sıfır değeri ile değiştirilir.

Return Value

İlgili grafikte bulunan en düşük değerin indeksi (geçerli çubuğa göre) veya bir hata durumunda -1. Hata detayları için,GetLastError() fonksiyonunu çağırın.

Örnek:

```
   double val;
//--- 15 ardışık çubuk arasında gerçek hacmin en düşük değerine sahip bir çubuk ara
//--- Şu andaki zaman dilimine göre endeks 10'dan endeks 24'e kadar.
   int val_index=iLowest(NULL,0,MODE_REAL_VOLUME,15,10);
   if(val_index!=-1)
      val=Low[val_index];
   else
      PrintFormat("iLowest() call error. Error code=%d",GetLastError());
```

iOpen

İlgili grafikteki çubuğun Açılış fiyatını ('shift' parametresiyle gösterilir) döndürür.

```
double  iOpen(
   const string         symbol,          // Sembol
   ENUM_TIMEFRAMES      timeframe,       // Periyot
   int                  shift            // Shift
   );
```

Parametreler

symbol

[in] Finansal enstrümanın sembol ismi. NULL şimdiki sembol anlamına gelir.

timeframe

[in] Periyot. ENUM_TIMEFRAMES sayısının değerlerinden biri olabilir. 0 şimdiki grafik periyodu anlamına gelir.

shift

[in] Zaman çizelgelerinden alınan değerin indeksi (mevcut çubuğa göre belirlenen sayıda çubuk ile geriye doğru kaydırma).

Return Value

Çubuğun açılış fiyatı ('shift' parametresi ile gösterilir) ilgili grafikte ya da bir hata durumunda 0. Hata detayları için,GetLastError() fonksiyonunu çağırın.

Not

Fonksiyon her zaman gerçek dataları döndürür. Bu amaçla, her bir arama sırasında belirtilen sembol/periyot için zaman çizelgesine bir istek yapar. Bu, ilk işlev çağrısı sırasında hazır bir veri yoksa, sonucu hazırlamak için biraz zaman alınabileceği anlamına gelir.

İşlev önceki arama sonuçlarını saklamaz ve hızlı değer döndürme için yerel önbellek yoktur.

Örnek:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Fonksiyon-olay işleyicisi "tik"                                  |
//+------------------------------------------------------------------+
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"
            );
  }
```

Ayrıca bakınız

CopyOpen, CopyRates

iTime

İlgili grafikteki çubuğun açılış zamanını ('shift' parametresiyle gösterilir) döndürür.

```
datetime  iTime(
   const string        symbol,          // Sembol
   ENUM_TIMEFRAMES     timeframe,       // Periyot
   int                 shift            // Shift
   );
```

Parametreler

symbol

[in] Finansal enstrümanın sembol ismi. NULL şimdiki sembol anlamına gelir.

timeframe

[in] Periyot. ENUM_TIMEFRAMES sayısının değerlerinden biri olabilir. 0 şimdiki grafik periyodu anlamına gelir.

shift

[in] Zaman çizelgelerinden alınan değerin endeksi (mevcut çubuğa göre belirlenen sayıda çubuk ile geriye doğru kaydırma).

Return Value

Çubuğun açılış zamanı ('shift' parametresi ile gösterilir) ilgili grafikte veya hata durumunda 0. Hata detayları içinGetLastError() fonksiyonunu çağırın.

Not

Fonksiyon her zaman gerçek verileri döndürür. Bu amaçla, her bir arama sırasında belirtilen sembol/periyot için zaman çizelgesine bir istek yapar. Bu, ilk işlev çağrısı sırasında hazır bir veri yoksa, sonucu hazırlamak için biraz zaman alınabileceği anlamına gelir.

İşlev önceki arama sonuçlarını saklamaz ve hızlı değer döndürme için yerel önbellek yoktur.

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Tarih Pazardır.
   datetime time=D'2018.06.10 12:00';
   string symbol="GBPUSD";
   ENUM_TIMEFRAMES tf=PERIOD_H1;
   bool exact=false;
//--- Belirtilen zamanda bir bar yok, iBarShift en yakın çubuğun dizinini döndürecekt
   int bar_index=iBarShift(symbol,tf,time,exact);
   PrintFormat("1. %s %s %s(%s): bar index is %d (exact=%s)",
               symbol,EnumToString(tf),TimeToString(time),DayOfWeek(time),bar_index,st
   datetime bar_time=iTime(symbol,tf,bar_index);
```

```
   PrintFormat("Time of bar #%d is %s (%s)",
               bar_index,TimeToString(bar_time),DayOfWeek(bar_time));
//PrintFormat(iTime(symbol,tf,bar_index));
//--- Çubuğun endeksini belirtilen süre ile isteyin; ama bar yok, dönüş -1
   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
  }
//+------------------------------------------------------------------+
//| Haftanın gününün adını döndürür                                  |
//+------------------------------------------------------------------+
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;
  }
/* Sonuç:
  1. GBPUSD PERIOD_H1 2018.06.10 12:00(SUNDAY): bar index 64 tür (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 -1 dir (exact=true)
*/
```

Ayrıca bakınız

CopyTime, CopyRates

iTickVolume

İlgili grafikteki çubuğun tik hacmini ('shift' parametresi ile gösterilir) döndürür.

```
long  iTickVolume(
   const string         symbol,          // Sembol
   ENUM_TIMEFRAMES      timeframe,       // Periyot
   int                  shift            // Shift
   );
```

Parametreler

symbol

[in] Finansal enstrümanın sembol ismi. NULL şimdiki sembol anlamına gelir.

timeframe

[in] Periyot. ENUM_TIMEFRAMES sayısının değerlerinden biri olabilir. 0 şimdiki grafik periyodu anlamına gelir.

shift

[in] Zaman serilerinden alınan değerin indeksi (mevcut çubuğa göre belirlenen sayıda çubuk ile geriye doğru kaydırma).

Return Value

İlgili grafikteki çubuğun tik hacmi ('shift' parametresi ile gösterilir) veya bir hata durumunda 0. Hata detayları için,GetLastError() fonksiyonunu çağırın.

Not

Fonksiyon her zaman gerçek verileri döndürür. Bu amaçla, her bir arama sırasında belirtilen sembol/periyot için zaman çizelgesine bir istek yapar. Bu, ilk işlev çağrısı sırasında hazır bir veri yoksa, sonucu hazırlamak için biraz zaman alınabileceği anlamına gelir.

İşlev önceki arama sonuçlarını saklamaz ve hızlı değer döndürme için yerel önbellek yoktur.

Örnek:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Fonksiyon-olay işleyicisi "tik"                                  |
//+------------------------------------------------------------------+
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"
            );
  }
```

Ayrıca bakınız

CopyTickVolume, CopyRates

iRealVolume

İlgili grafikteki çubuğun gerçek hacmini ('shift' parametresiyle gösterilir) döndürür.

```
long  iRealVolume(
   const string         symbol,          // Sembol
   ENUM_TIMEFRAMES      timeframe,       // Periyot
   int                  shift            // Shift
   );
```

Parametreler

symbol

[in] Finansal enstrümanın sembol ismi. NULL şimdiki sembol anlamına gelir.

timeframe

[in] Periyot. ENUM_TIMEFRAMES sayısının değerlerinden biri olabilir. 0 şimdiki grafik periyodu anlamına gelir.

shift

[in] Zaman çizelgelerinden alınan değerin indeksi (mevcut çubuğa göre belirlenen sayıda çubuk ile geriye doğru kaydırma).

Return Value

İlgili grafikteki çubuğun gerçek hacmi ('shift' parametresi ile gösterilir) veya bir hata durumunda 0. Hata detayları için,GetLastError() fonksiyonunu çağırın.

Not

Fonksiyon her zaman gerçek verileri döndürür. Bu amaçla, her bir arama sırasında belirtilen sembol/periyot için zaman çizelgesine bir istek yapar. Bu, ilk işlev çağrısı sırasında hazır bir veri yoksa, sonucu hazırlamak için biraz zaman alınabileceği anlamına gelir.

İşlev önceki arama sonuçlarını saklamaz ve hızlı değer döndürme için yerel önbellek yoktur.

Örnek:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Fonksiyon-olay işleyicisi "tik"                                  |
//+------------------------------------------------------------------+
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"
            );
  }
```

Ayrıca bakınız

CopyRealVolume, CopyRates

iVolume

İlgili grafikteki çubuğun tik hacmini ('shift' parametresi ile gösterilir) döndürür.

```
long  iVolume(
   const string         symbol,          // Sembol
   ENUM_TIMEFRAMES      timeframe,       // Periyot
   int                  shift            // Shift
   );
```

Parametreler

symbol

[in] Finansal enstrümanın sembol ismi. NULL şimdiki sembol anlamına gelir.

timeframe

[in] Periyot. ENUM_TIMEFRAMES numaralandırmanın değerlerinden biri olabilir. 0 şimdiki grafik periyodu anlamına gelir.

shift

[in] Zaman çizelgelerinden alınan değerin indeksi (mevcut çubuğa göre belirlenen sayıda çubuk ile geriye doğru kaydırma).

Return Value

İlgili grafikteki çubuğun tik hacmi ('shift' parametresi ile gösterilir) veya bir hata durumunda 0. Hata detayları içinGetLastError() fonksiyonunu çağırın.

Not

Fonksiyon her zaman gerçek verileri döndürür. Bu amaçla, her bir arama sırasında belirtilen sembol/periyot için zaman çizelgesine bir istek yapar. Bu, ilk işlev çağrısı sırasında hazır bir veri yoksa, sonucu hazırlamak için biraz zaman alınabileceği anlamına gelir.

İşlev önceki arama sonuçlarını saklamaz ve hızlı değer döndürme için yerel önbellek yoktur.

Örnek:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Fonksiyon-olay işleyicisi "tik"                                  |
//+------------------------------------------------------------------+
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"
            );
  }
```

Ayrıca bakınız

CopyTickVolume, CopyRates

iSpread

İlgili grafikteki çubuğun spread değerini ('shift' parametresiyle gösterilir) döndürür.

```
long  iSpread(
   const string         symbol,          // Sembol
   ENUM_TIMEFRAMES      timeframe,       // Periyot
   int                  shift            // Shift
   );
```

Parametreler

symbol

[in] Finansal enstrümanın sembol ismi. NULL şimdiki sembol anlamına gelir.

timeframe

[in] Periyot. ENUM_TIMEFRAMES sayısının değerlerinden biri olabilir. 0 şimdiki grafik periyodu anlamına gelir.

shift

[in] Zaman çizelgelerinden alınan değerin indeksi (mevcut çubuğa göre belirlenen sayıda çubuk ile geriye doğru kaydırma).

Return Value

İlgili grafikteki çubuğun ('shift' parametresi ile gösterilir) spread değeri veya bir hata durumunda 0. Hata detayları için,GetLastError() fonksiyonunu çağırın.

Not

Fonksiyon her zaman gerçek verileri döndürür. Bu amaçla, her bir arama sırasında belirtilen sembol/periyot için zaman çizelgesine bir istek yapar. Bu, ilk işlev çağrısı sırasında hazır bir veri yoksa, sonucu hazırlamak için biraz zaman alınabileceği anlamına gelir.

İşlev önceki arama sonuçlarını saklamaz ve hızlı değer döndürme için yerel önbellek yoktur.

Örnek:

```
input int shift=0;
//+------------------------------------------------------------------+
//| Fonksiyon-olay işleyicisi "tik"                                  |
//+------------------------------------------------------------------+
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"
            );
  }
```

Ayrıca bakınız

CopySpread, CopyRates

Kullanıcı-tanımlı semboller

Kullanıcı-tanımlı sembollerin özelliklerini oluşturmak ve değiştirmek için kullanılan fonksiyonlar.

Kullanıcılar, terminal ile belli bir alım-satım sunucusuna bir kez bağlandıklarında, aracı kurum tarafından sağlanan finansal sembollerin zaman-serileri ile çalışabilirler. Kullanılabilecek mevcut sembollerin bir listesi Piyasa Gözlemi penceresinde gösterilir. Ayrı bir fonksiyonlar grubu bu sembollerin özelliklerine dair verilerin, işlem seansı verilerinin ve piyasa derinliği güncellemelerinin alınmasını sağlar.

Bu bölümde açıklanan fonksiyonlar kullanıcı-tanımlı sembollerle çalışmak için tasarlanmıştır. Sembol oluşturmak için alım-satım sunucusunda mevcut olan sembollerin verilerini kullanabileceğiniz gibi, metin dosyalarını ve dışsal veri kaynaklarını da kullanabilirsiniz.

Fonksiyon	Eylem
CustomSymbolCreate	Belirtilen grubun içinde belirtilen isimde bir sembol oluşturur
CustomSymbolDelete	Belirtilen isme sahip olan kullanıcı-tanımlı sembolü siler
CustomSymbolSetInteger	Kullanıcı-tanımlı sembolün tamsayı tipli bir özelliğini ayarlar
CustomSymbolSetDouble	Kullanıcı-tanımlı sembolün reel tipli bir özelliğini ayarlar
CustomSymbolSetString	Kullanıcı-tanımlı sembolün dizgi tipli bir özelliğini ayarlar
CustomSymbolSetMarginRate	Kullanıcı-tanımlı bir sembol için emir türüne ve yönüne göre ilgili teminat oranlarını ayarlar
CustomSymbolSetSessionQuote	Belirtilen sembol ve gün değeri için fiyatlandırma seansının başlangıç ve bitiş zamanlarını ayarlar
CustomSymbolSetSessionTrade	Belirtilen sembol ve gün değeri için işlem seansının başlangıç ve bitiş zamanlarını ayarlar
CustomRatesDelete	Kullanıcı-tanımlı sembolün fiyat geçmişi üzeride belirtilen zaman aralığındaki tüm çubuk verilerini siler.
CustomRatesReplace	Kullanıcı-tanımlı sembolün veri geçmişinde, belirtilen zaman aralığındaki verileri 'MqlRates' tipli dizinin verileri ile değiştirir
CustomRatesUpdate	Kullanıcı-tanımlı bir sembolün veri geçmişindeki eksik verileri ekler ve mevcut verileri 'MqlRates' tipli diziden kopyalayarak değiştirir
CustomTicksAdd	Kullanıcı-tanımlı bir sembol için MqlTick tipli bir diziden veri ekler. Kullanıcı-tanımlı sembol Piyasa Gözlemi penceresinde seçilmiş olmalıdır
CustomTicksDelete	Kullanıcı-tanımlı sembolün belirtilen aralıktaki geçmiş tik verilerini siler
CustomTicksReplace	Kullanıcı-tanımlı sembolün belirtilen zaman aralığındaki geçmiş verilerini 'MqlTick' tipli diziden alınan verilerle değiştirir
CustomBookAdd	Kullanıcı-tanımlı sembol için Piyasa Derinliği durumunu aktarır

CustomSymbolCreate

Söz konusu grupta belirtilen ada sahip özel bir sembol oluşturur.

```
bool  CustomSymbolCreate(
   const string      symbol_name,          // özel sembol adı
   const string      symbol_path="",       // sembolün içinde oluşacağı grubun adı
   const string      symbol_origin=NULL    // özel bir sembol oluşturmak için temel ola
   );
```

Parametreler

symbol_name

[in] Özel sembol adı. Sembolün içinde bulunduğu grupları veya alt grupları içermemelidir.

symbol_path=""

[in] Sembolün bulunduğu grubun ismi.

symbol_origin=NULL

[in] Oluşturulan özel sembole özelliklerinin kopyalanacağı sembolün adı. Özel bir sembol oluşturulduktan sonra, uygun fonksiyonlar kullanılarak herhangi bir özellik değeri gerekli bir değerle değiştirilebilir.

Geri dönüş değeri

true - başarı, aksi takdirde - false. Hata hakkında bilgi edinmek için, GetLastError() fonksiyonunu çağırın.

Not

Tüm özel semboller farklı Özel bölümde oluşturulur. Bir grup adı belirtilmezse (CustomSymbolCreate fonksiyonunda *symbol_path* parametresi boş bir dize veya NULL içeriyorsa), Özel bölüm kökünde özel bir sembol oluşturulur. Burada, grupların ve alt grupların klasör ve alt klasörler olarak görüntülenebildiği dosya sistemi ile bir benzetme yapabiliriz

Sembol ve grup adları; noktalama işaretleri, boşluklar veya özel karakterler içermeyen Latin harflerini içerebilir (yalnızca ".", "_", "&" ve "#" içerebilir) <, >, :, ", /, |, ?, * karakterlerinin kullanılması önerilmez.

Özel sembol adı, içinde oluşturulduğu grubun adına bakılmaksızın benzersiz olmalıdır. Aynı ada sahip bir sembol zaten mevcutsa, CustomSymbolCreate() fonksiyonu 'false' değerini geri döndürürken, bir sonraki GetLastError() çağrısı 5300 hatasını (ERR_NOT_CUSTOM_SYMBOL) veya 5304 hatasını (ERR_CUSTOM_SYMBOL_EXIST) geri döndürür.

Sembol adının uzunluğu 31 karakteri geçmemelidir. Aksi takdirde; CustomSymbolCreate(), 'false' geri değerini döndürür ve 5302 - ERR_CUSTOM_SYMBOL_NAME_LONG hatasını etkinleştirir.

symbol_path parametresi iki şekilde ayarlanabilir:

- isimsiz bir özel sembolün içinde olduğu grubun sadece bir ismi, örneğin - "CFD\\Metals" Hataları önlemek için bu seçeneği kullanmak en iyisidir.
- veya <grup> ismi + grup ayırıcı "\\"+<özel sembol ismi>, örneğin - "CFD\\Metals\\Platinum". Bu durumda, grup adı özel sembolün tam adı ile bitmelidir. Uyumsuzluk durumunda, özel sembol hala oluşturulur, ancak amaçlanan grupta oluşturulmaz. Örneğin, eğer *symbol_path*="CFD\\Metals\\Platinum" ve *symbol_name*="platinum" (kayıt hatası), bu durumda

"Custom\CFD\Metals\Platinum" grubunda "platinum" isimli bir özel sembol oluşturulur. SymbolInfoGetString ("platinum", SYMBOL_PATH) fonksiyonu "Custom\CFD\Metals\Platinum\platinum" değerini geri döndürür.

SYMBOL_PATH özelliğinin sonunda sembol adının bulunduğu yolu geri döndürür. Bu nedenle, birebir aynı grupta özel bir sembol oluşturmak istiyorsanız, değişiklik yapılmadan kopyalanamaz. Bu durumda, yukarıda açıklanan sonucu elde etmemek için sembol adını kesmek gerekir.

Eğer var olmayan bir sembol *symbol_origin* parametresi olarak ayarlanmışsa; özel sembol, *symbol_origin* parametresi ayarlanmamış gibi boş olarak oluşturulur. Bu durumda, 4301 - ERR_MARKET_UNKNOWN_SYMBOL hatası etkinleşir.

symbol_path parametre uzunluğu, "Custom\\", "\\" grup ayırıcıları ve sonunda belirtilmişse sembol adını dikkate alarak 127 karakteri geçmemelidir.

Ayrıca bakınız

SymbolName, SymbolSelect, CustomSymbolDelete

CustomSymbolDelete

Belirtilen isme sahip olan kullanıcı-tanımlı sembolü siler.

```
bool  CustomSymbolDelete(
   const string      symbol_name          // kullanıcı-tanımlı sembolün ismi
   );
```

Parametreler

symbol

[in] Kullanıcı-tanımlı sembolün ismi Var olan bir sembolün ismiyle örtüşmemelidir.

Dönüş Değeri

Başarı durumunda 'true', aksi durumda 'false'. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

Not

Çizelgesi açık durumda olan veya Piyasa gözleminde gösterilen kullanıcı-tanımlı semboller silinemez.

Ayrıca bakınız

SymbolName, SymbolSelect, CustomSymbolCreate

CustomSymbolSetInteger

Kullanıcı-tanımlı sembolün tamsayı tipli bir özelliğini ayarlar.

```
bool  CustomSymbolSetInteger(
   const string               symbol_name,      // sembol ismi
   ENUM_SYMBOL_INFO_INTEGER   property_id,      // özelliğin tanımlayıcısı
   long                       property_value    // özelliğin değeri
   );
```

Parametreler

symbol_name

[in] Kullanıcı-tanımlı sembolün ismi

property_id

[in] Sembol özelliğinin tanımlayıcısı. Değer, ENUM_SYMBOL_INFO_INTEGER sayımının değerlerinden biri olabilir.

property_value

[in] Özellik değerini taşıyan 'long' tipli değişken.

Dönüş Değeri

Başarı durumunda 'true', aksi durumda 'false'. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

Not

Sembolün dakika ve tik geçmiş kayıtları tamamen kaldırılır eğer bu özelliklerden herhangi biri sembol spesifikasyonunda değiştirilse:

- SYMBOL_CHART_MODE - barın oluşumu için fiyat türü (Teklif (Bid) veya Son (Last) fiyat)
- SYMBOL_DIGITS - fiyatı görüntülemek için ondalık noktadan sonraki basamak sayısı

Sembol geçmişini sildikten sonra terminal, güncellenmiş özellikleri kullanarak yeni bir geçmişi oluşturmaya çalışır. Aynı şey, sembol özellikleri manuel olarak değiştirildiğinde gerçekleşir.

Ayrıca bakınız

SymbolInfoInteger

CustomSymbolSetDouble

Kullanıcı-tanımlı sembolün reel tipli özelliğini ayarlar.

```
bool  CustomSymbolSetDouble(
   const string              symbol_name,      // sembol ismi
   ENUM_SYMBOL_INFO_DOUBLE   property_id,      // özelliğin tanımlayıcısı
   double                    property_value    // özelliğin değeri
   );
```

Parametreler

symbol_name

[in] Kullanıcı-tanımlı sembolün ismi

property_id

[in] Sembol özelliğinin tanımlayıcısı. Değer, ENUM_SYMBOL_INFO_DOUBLE sayımının değerlerinden biri olabilir.

property_value

[in] Özellik değerini taşıyan 'double' tipli değişken.

Dönüş Değeri

Başarı durumunda 'true', aksi durumda 'false'. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

Not

Sembolün dakika ve tik geçmiş kayıtları tamamen kaldırılır eğer bu özelliklerden herhangi biri sembol spesifikasyonunda değiştirilse:

- SYMBOL_POINT - bir nokta değeri
- SYMBOL_TRADE_TICK_SIZE - İzin verilen minimum fiyat değişikliğini belirten bir tik değeri
- SYMBOL_TRADE_TICK_VALUE - karlı bir pozisyon için tek tikli fiyat değişim değeri

Sembol geçmişini sildikten sonra terminal, güncellenmiş özellikleri kullanarak yeni bir geçmişi oluşturmaya çalışır. Aynı şey, sembol özellikleri manuel olarak değiştirildiğinde gerçekleşir.

Ayrıca bakınız

SymbolInfoDouble

CustomSymbolSetString

Kullanıcı-tanımlı sembolün dizgi tipli özelliğini ayarlar.

```
bool  CustomSymbolSetString(
   const string              symbol_name,      // sembol ismi
   ENUM_SYMBOL_INFO_STRING   property_id,      // özelliğin tanımlayıcısı
   string                    property_value    // özelliğin değeri
   );
```

Parametreler

symbol_name

[in] Kullanıcı-tanımlı sembolün ismi

property_id

[in] Sembol özelliğinin tanımlayıcısı. Bu değer, ENUM_SYMBOL_INFO_STRING sayımının değerlerinden biri olabilir.

property_value

[in] Özellik değerini taşıyan dizgi tipli değişken.

Dönüş Değeri

Başarı durumunda 'true', aksi durumda 'false'. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

Not

Eğer SYMBOL_FORMULA özelliği (sembol fiyat yapımı için denklemi ayarlama) sembol spesifikasyonunda değiştirilirse, sembolün dakika ve tik geçmiş kayıtları tamamen kaldırılır. Sembol geçmiş kayıdı silindikten sonra terminal, yeni denklemi kullanarak yeni bir geçmişi yaratmaya çalışır. Aynı şey, sembol denklemi manuel olarak değiştirildiğinde gerçekleşir.

Ayrıca bakınız

SymbolInfoString

CustomSymbolSetMarginRate

Kullanıcı-tanımlı bir sembol için emir türüne ve yönüne göre ilgili teminat oranlarını ayarlar.

```
bool  CustomSymbolSetMarginRate(
   const string       symbol_name,              // sembol ismi
   ENUM_ORDER_TYPE    order_type,               // emir tipi
   double             initial_margin_rate,      // başlangıç teminat oranı
   double             maintenance_margin_rate   // sürdürme teminatı oranı
   );
```

Parametreler

symbol_name

[in] Kullanıcı-tanımlı sembolün ismi

order_type

[in] Emir tipi.

initial_margin_rate

[in] Başlangıç teminat oranını taşıyan double tipli değişken. Başlangıç teminatı, belli bir yönde 1 lotluk işlem yapmak için gereken mevduat miktarıdır. İlgili oranının başlangıç teminatı ile çarpımı, belirtilen türde bir emir için hesaptaki mevduatın ne kadarının rezerve edileceğini gösterir.

maintenance_margin_rate

[in] Sürdürme teminatı oranını taşıyan double tipli değişken. Sürdürme teminatı, belli bir yöndeki 1 lotluk pozisyonu açık tutmak için gereken minimum mevduat miktarıdır. İlgili oranının sürdürme teminatı ile çarpımı, belirtilen türde bir emrin gerçekleştirilmesinden sonra hesaptaki mevduatın ne kadarının rezerve edileceğini gösterir.

Dönüş Değeri

Başarı durumunda 'true', aksi durumda 'false'. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

Ayrıca bakınız

SymbolInfoMarginRate

CustomSymbolSetSessionQuote

Belirtilen sembol ve gün değeri için fiyatlandırma seansının başlangıç ve bitiş zamanlarını ayarlar.

```
bool  CustomSymbolSetSessionQuote(
   const string      symbol_name,          // sembol ismi
   ENUM_DAY_OF_WEEK  day_of_week,          // haftanın günü
   uint              session_index,        // seans indisi
   datetime          from,                 // seans başlangıç zamanı
   datetime          to                    // seans bitiş zamanı
   );
```

Parametreler

symbol_name

[in] Kullanıcı-tanımlı sembolün ismi

ENUM_DAY_OF_WEEK

[in] Haftanın günü. ENUM_DAY_OF_WEEK sayımının değerlerinden biri olabilir.

uint

[in] Başlangıç ve bitiş zamanlarının ayarlanacağı seansın indisi. Seansların indislenmesine '0' ile başlanır.

from

[in] 00:00 şeklinde, saniye bazında seans başlangıç zamanı; değişkenin veri değeri gözardı edilir.

to

[in] 00:00 şeklinde, saniye bazında seans bitiş zamanı; değişkenin veri değeri gözardı edilir.

Dönüş Değeri

Başarı durumunda 'true', aksi durumda 'false'. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

Not

Belirtilen *session_index* değerine karşılık gelen seans zaten mevcutsa fonksiyon ilgili başlangıç ve bitiş değerlerini basitçe değiştirir.

Başlangıç ve bitiş parametrelerine '0' değeri geçirilmişse (*from=0* ve *to=0*), *session_index* değerine karşılık gelen seans silinir ve takip eden seans indisleri bir aşağı kaydırılır.

Seanslar sadece ardışık sırayla eklenebilir. Yani, *session_index=1* değerine karşılık gelen seans, sadece 0 indisli seans mevcutsa eklenebilir. Bu kural çiğnendiğinde yeni seans oluşturulmaz ve fonksiyon 'false' dönüşü yapar.

Ayrıca bakınız

SymbolInfoSessionQuote, Sembol bilgisi, TimeToStruct, Tarih yapısı

CustomSymbolSetSessionTrade

Belirtilen sembol ve gün değeri için işlem seansının başlangıç ve bitiş zamanlarını ayarlar.

```
bool  CustomSymbolSetSessionTrade(
   const string      symbol_name,          // sembol ismi
   ENUM_DAY_OF_WEEK  day_of_week,          // haftanın günü
   uint              session_index,        // seans indisi
   datetime          from,                 // seans başlangıç zamanı
   datetime          to                    // seans bitiş zamanı
   );
```

Parametreler

symbol_name

[in] Kullanıcı-tanımlı sembolün ismi

ENUM_DAY_OF_WEEK

[in] Haftanın günü. ENUM_DAY_OF_WEEK sayımının değerlerinden biri olabilir.

uint

[in] Başlangıç ve bitiş zamanlarının ayarlanacağı seansın indisi. Seansların indislenmesine '0' ile başlanır.

from

[in] 00:00 şeklinde, saniye bazında seans başlangıç zamanı; değişkenin veri değeri gözardı edilir.

to

[in] 00:00 şeklinde, saniye bazında seans bitiş zamanı; değişkenin veri değeri gözardı edilir.

Dönüş Değeri

Başarı durumunda 'true', aksi durumda 'false'. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

Not

Belirtilen *session_index* değerine karşılık gelen seans zaten mevcutsa fonksiyon ilgili başlangıç ve bitiş değerlerini basitçe değiştirir.

Başlangıç ve bitiş parametrelerine '0' değeri geçirilmişse (*from=0* ve *to=0*), *session_index* değerine karşılık gelen seans silinir ve takip eden seans indisleri bir aşağı kaydırılır.

Seanslar sadece ardışık sırayla eklenebilir. Yani, *session_index=1* değerine karşılık gelen seans, sadece 0 indisli seans mevcutsa eklenebilir. Bu kural çiğnendiğinde yeni seans oluşturulmaz ve fonksiyon 'false' dönüşü yapar.

Ayrıca bakınız

SymbolInfoSessionTrade, Sembol bilgisi, TimeToStruct, Tarih yapısı

CustomRatesDelete

Kullanıcı-tanımlı sembolün fiyat geçmişi üzeride belirtilen zaman aralığındaki tüm verileri siler.

```
int  CustomRatesDelete(
   const string      symbol,        // sembol ismi
   datetime          from,          // başlangıç tarihi
   datetime          to             // bitiş tarihi
   );
```

Parametreler

symbol

[in] Kullanıcı-tanımlı sembolün ismi

from

[in] Fiyat geçmişi üzerinde silinmesi istenilen veri aralığının başlangıç zamanı.

to

[in] Fiyat geçmişi üzerinde silinmesi istenilen veri aralığının bitiş zamanı.

Dönüş Değeri

Silinene çubukların sayısı veya hata durumunda '-1'.

Ayrıca bakınız

CustomRatesReplace, CustomRatesUpdate, CopyRates

CustomRatesReplace

Kullanıcı-tanımlı sembolün veri geçmişinde, belirtilen zaman aralığındaki verileri MqlRates tipli dizinin verileri ile değiştirir.

```
int  CustomRatesReplace(
   const string     symbol,              // sembol ismi
   datetime         from,                // başlangıç tarihi
   datetime         to,                  // bitiş tarihi
   const MqlRates&  rates[],             // kullanıcı-tanımlı sembol için kullanılacak
   uint             count=WHOLE_ARRAY    // kullanılacak rates[] dizisi elemanlarının s
   );
```

Parametreler

symbol

[in] Kullanıcı-tanımlı sembolün ismi

from

[in] Fiyat geçmişinde değiştirilmek istenilen veri aralığının başlangıç zamanı.

to

[in] Fiyat geçmişinde değiştirilmek istenilen veri aralığının bitiş zamanı.

rates[]

[in] M1 için MqlRates tipli geçmiş veri dizisi.

count=WHOLE_ARRAY

[in] Değişim için kullanılacak *rates[]* dizisi elemanlarının sayısı. WHOLE_ARRAY, tüm *rates[]* dizisi elemanlarının değişim için kullanılması gerektiği anlamına gelir.

Geri Dönüş Değeri

Güncellenen veri sayısı veya hata durumunda '-1'.

Not

rates[] dizisinde belirtilen aralıkta olmayan veriler gözardı edilir. Fiyat geçmişinin belirtilen aralığında mevcut olan veriler yenileriyle değiştirilir. Belirtilen zaman aralığının dışında kalan veriler ise değiştirilmeden bırakılır. *rates[]* dizisindeki veriler OHLC fiyatlarına göre doğru olmalıdır ve çubuk açılış zamanları M1 zaman dilimine karşılık gelmelidir.

Ayrıca bakınız

CustomRatesDelete, CustomRatesUpdate, CopyRates

CustomRatesUpdate

Kullanıcı-tanımlı bir sembolün veri geçmişindeki eksik verileri ekler ve mevcut verileri MqlRates tipli diziden kopyalayarak değiştirir.

```
int  CustomRatesUpdate(
   const string     symbol,             // kullanıcı-tanımlı sembolün ismi
   const MqlRates&  rates[],            // kullanıcı-tanımlı sembol için kullanılacak
   uint             count=WHOLE_ARRAY   // kullanılacak rates[] dizisi elemanlarının s
   );
```

Parametreler

symbol

[in] Kullanıcı-tanımlı sembolün ismi

rates[]

[in] M1 için MqlRates tipli geçmiş veri dizisi.

count=WHOLE_ARRAY

[in] Güncelleme için kullanılacak *rates[]* dizisi elemanlarının sayısı. WHOLE_ARRAY, tüm *rates[]* dizisi elemanlarının kullanılması gerektiği anlamına gelir.

Geri Dönüş Değeri

Güncellenen veri sayısı veya hata durumunda '-1'.

Not

Kullanıcı-tanımlı sembolün *rates[]* dizisinde hiç çubuk verisi yoksa, eklenir. Çubuk verisi mevcutsa, değiştirilir. Mevcut fiyat geçmişindeki diğer tüm veriler değiştirilmeden bırakılır. *rates[]* dizisindeki veriler OHLC fiyatlarına göre doğru olmalıdır ve çubuk açılış zamanları M1 zaman dilimine karşılık gelmelidir.

Ayrıca bakınız

CustomRatesReplace, CustomRatesDelete, CopyRates

CustomTicksAdd

Kullanıcı-tanımlı bir sembol için MqlTick tipli bir diziden veri ekler. Kullanıcı-tanımlı sembol Piyasa Gözlemi penceresinde seçilmiş olmalıdır.

```
int  CustomTicksAdd(
   const string     symbol,              // Sembol ismi
   const MqlTick&   ticks[],             // Kullanıcı-tanımlı sembole yüklenebilecek ti
   uint             count=WHOLE_ARRAY    // kullanılacak ticks[] dizisi elemanlarının s
   );
```

Parametreler

symbol

[in] Kullanıcı-tanımlı sembolün ismi.

ticks[]

[in] Geçmişten şimdiye doğru düzenlenmiş (k<n ise ticks[k].time_msc <= ticks[n].time_msc) verileri içeren MqlTick tipli dizi.

count=WHOLE_ARRAY

[in] Ekleme için kullanılacak *ticks[]* dizisi elemanlarının sayısı. WHOLE_ARRAY, tüm *ticks[]* dizisi elemanlarının kullanılması gerektiği anlamına gelir.

Geri Dönüş Değeri

Eklenen tik verilerinin sayısı veya hata durumunda '-1'.

Notlar

CustomTicksAdd fonksiyonu sadece Piyasa Gözlemi penceresinde seçili olan kullanıcı-tanımlı sembol için çalışır. Sembol Piyasa Gözleminde seçili değilse, veri eklemek için CustomTicksReplace fonksiyonunu kullanmalısınız.

CustomTicksAdd fonksiyonu, tikleri broker'ın serverından iletmeye izin verir. Veriler doğrudan tik veritabanına yazılmak yerine Market İzleme penceresine gönderilir. Terminal daha sonra Market İzlemeden tikleri bir veritabanına kaydeder. Bir fonksiyon çağrısı sırasında iletilen veri miktarı büyükse, fonksiyonun davranışı, kaynak kullanımını azaltmak için değişir. Eğer 256'dan fazla tik oluşmuşssa, data 2 parçaya bölünür. İlki, örneğin büyük parça direkt olarak tik veritabanına yazdırılır (CustomTicksReplace'de olduğu gibi). 128 tiki içeren ikinci parça, terminalin verileri bir veritabanına kaydettiği yerden Market İzleme penceresine geçirilir.

MqlTick yapısının zaman değerli iki alanı vardır: time (saniye cinsinden tik zamanı) ve time_msc (milisaniye cinsinden tik zamanı). Bunlar, 1 Ocak 1970'den itibaren sıralanmıştır. Eklenen tikler için bu alanlar şu sırayla işlenir:

1. ticks[k].time_msc!=0 ise, veri ticks[k].time alanını doldurmak için kullanılır, yani ticks[k].time=ticks[k].time_msc/1000 (tamsayı bölümü) tik verisine uygulanır
2. ticks[k].time_msc==0 ve ticks[k].time!=0 ise, milisaniye cinsi zaman değeri 1000 ile çarpılarak elde edilir (ticks[k].time_msc=ticks[k].time*1000)
3. ticks[k].time_msc==0 ve ticks[k].time==0 ise, milisaniye cinsinden alım-satım sunucusunun zamanı CustomTicksAdd çağrısı yapıldığı an bu alanlara yazılır.

ticks[k].bid, ticks[k].ask, ticks[k].last veya ticks[k].volume sıfırdan büyükse, ticks[k].flags alanına ilgili bayrakların uygun bir kombinasyonu yazılır

- TICK_FLAG_BID - tik verisi satış fiyatını değiştirdi
- TICK_FLAG_ASK - tik verisi alış fiyatını değiştirdi
- TICK_FLAG_LAST - tik verisi son işlem fiyatını değiştirdi
- TICK_FLAG_VOLUME - tik verisi hacmi değiştirdi

Alanlardan birinin değeri sıfır veya daha küçükse, karşılık gelen bayraklar ticks[k].flags alanına yazılmaz.

TICK_FLAG_BUY ve TICK_FLAG_SELL bayrakları kullanıcı-tanımlı sembol geçmişine eklenmez.

Ayrıca bakınız

CustomRatesDelete, CustomRatesUpdate, CustomTicksReplace, CopyTicks, CopyTicksRange

CustomTicksDelete

Kullanıcı-tanımlı sembolün belirtilen aralıktaki geçmiş verilerini siler.

```
int  CustomTicksDelete(
   const string      symbol,            // sembol ismi
   long              from_msc,          // başlangıç tarihi
   long              to_msc             // bitiş tarihi
   );
```

Parametreler

symbol

[in] Kullanıcı-tanımlı sembolün ismi

from_msc

[in] Değiştirilmek istenilen veri aralığının başlangıç zamanı. Zaman 01.01.1970'den itibaren milisaniyeler biçiminde olmalıdır.

to_msc

[in] Değiştirilmek istenilen veri aralığının son zamanı. Zaman 01.01.1970'den itibaren milisaniyeler biçiminde olmalıdır.

Dönüş Değeri

Silinene tik verilerinin sayısı veya hata durumunda '-1'.

Ayrıca bakınız

CustomRatesDelete, CustomRatesUpdate, CustomTicksReplace, CopyTicks, CopyTicksRange

CustomTicksReplace

Kullanıcı-tanımlı sembolün belirtilen zaman aralığındaki geçmiş verilerini MqlTick tipli diziden alınan verilerle değiştirir.

```
int  CustomTicksReplace(
   const string     symbol,            // sembol ismi
   long             from_msc,          // başlangıç tarihi
   long             to_msc,            // bitiş tarihi
   const MqlTick&   ticks[],           // kullanıcı-tanımlı sembol için kullanılacak v
   uint             count=WHOLE_ARRAY  // kullanılacak ticks[] dizisi elemanlarının sa
   );
```

Parametreler

symbol

[in] Kullanıcı-tanımlı sembolün ismi

from_msc

[in] Değiştirilmek istenilen veri aralığının başlangıç zamanı. Zaman 01.01.1970'den itibaren milisaniyeler biçiminde olmalıdır.

to_msc

[in] Değiştirilmek istenilen veri aralığının son zamanı. Zaman 01.01.1970'den itibaren milisaniyeler biçiminde olmalıdır.

ticks[]

[in] Zamana göre artan sırayla MqlTick tipli veri dizisi.

count=WHOLE_ARRAY

[in] Belirtilen zaman aralığında değişim için kullanılacak *ticks[]* dizisi elemanlarının sayısı. WHOLE_ARRAY, tüm *ticks[]* dizisi elemanlarının kullanılması gerektiği anlamına gelir.

Geri Dönüş Değeri

Güncellenen veri sayısı veya hata durumunda '-1'.

Not

Fiyat akışı içinde aynı zaman indisine sahip birkaç tik verisi bulunabileceği için (kesin zaman değeri MqlTick yapısının *time_msc* alanında yer alır), CustomTicksReplace fonksiyonu *ticks[]* dizisinin elemanlarını otomatik olarak sıralamaz. Bu nedenle, dizideki tikler önceden, zamana göre artan şekilde sıralanmalıdır.

Tik verileri, *to_msc* değerine ulaşılıncaya kadar veya hata oluşuncaya kadar zamana göre birer birer değiştirilir. Tik verisinin zamanı ve artan sıra yapısı arasında bir uyuşmazlık tespit edilirse, veri değiştirme işlemi orada sonlandırılır. Belirtilen aralıkta, uyuşmazlık noktasından önceki veriler başarıyla değiştirilir, fakat o noktadan sonraki tarihlere ait veriler değiştirilmeden bırakılır.

ticks[] dizisi belli bir zaman aralığı için herhangi bir veri içermiyorsa, kullanıcı-tanımlı sembolün veri geçmişi üzerinde `ticks[]` dizisiyle değiştirilen ilgili alanlarda bir "boşluk" oluşur. Diğer bir deyişle, eksik verile CustomTicksReplace fonksiyonunun çağrılması, tıpkı CustomTicksDelete fonksiyonunda olduğu gibi, ilgili veri aralığını silecektir.

Eğer tik veri tabanı belirli bir zaman aralığı için hiçbir veri içermiyorsa, CustomTicksReplace ticks[] dizisinden tikleri veritabanına ekleyecektir.

CustomTicksReplace fonksiyonu tick veritabanı ile direkt olarak çalışır.

Ayrıca bakınız

CustomRatesDelete, CustomRatesUpdate, CustomTicksDelete, CopyTicks, CopyTicksRange

CustomBookAdd

Kullanıcı-tanımlı sembol için Piyasa Derinliği durumunu aktarır. Fonksiyon, fiyatlar bir broker sunucusundan geliyormuş gibi Piyasa Derinliğini yayınlamayı sağlar.

```
bool  CustomBookAdd(
   const string        symbol,            // sembol adı
   const MqlBookInfo&  books[]            // Piyasa Derinliği elemanlarının açıklamala
   uint                count=WHOLE_ARRAY  // kullanılacak eleman sayısı
   );
```

Parametreler

symbol

[in] Özel sembol adı.

books[]

[in] Piyasa Derinliği durumunu tam olarak tanımlayan MqlBookInfo dizi verisi - tüm alım ve satım istekleri. Aktarılan Piyasa Derinliği durumu tamamen bir öncekinin yerine geçer.

count=WHOLE_ARRAY

[in] Fonksiyona aktarılacak 'books' dizisi elemanlarının sayısı. Tüm dizi varsayılan olarak kullanılır.

Geri dönüş değeri

true - başarı, aksi takdirde - false. Hata hakkında bilgi edinmek için, GetLastError() fonksiyonunu çağırın.

Not

CustomBookAdd fonksiyonu sadece Piyasa Derinliğinin açıldığı özel sembollerde çalışır — platform arayüzü üzerinden veya MarketBookAdd fonksiyonu aracılığıyla.

Piyasa derinliği eklenirken, sembolün Alış ve Satış fiyatları güncellenmez. En iyi fiyatların değişimini kontrol etmelisiniz ve tikleri CustomTicksAdd fonksiyonunu kullanarak ilave etmelisiniz.

Fonksiyon, iletilen verilerin doğruluğunu kontrol eder: her öğe için tür, fiyat ve hacim belirtilmelidir. Ayrıca, MqlBookInfo.volume ve MqlBookInfo.volume_real sıfır veya negatif olmamalıdır; her iki hacim de negatifse, bu bir hata olarak kabul edilecektir. Hacimlerden herhangi birini veya her ikisini belirtebilirsiniz: belirtilen veya pozitif olan kullanılacaktır:

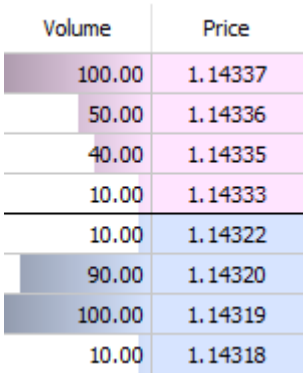
```
volume=-1 && volume_real=2 — volume_real=2 kullanılacaktır,
volume=3 && volume_real=0 — volume=3 kullanılacaktır.
```

MqlBookInfo elemanlarının 'books' dizisindeki sırası önemli değildir. Verileri kaydederken, terminal bunları fiyata göre sıralar.

Verileri kaydederken, alıcı özel sembolün "Book depth" (SYMBOL_TICKS_BOOKDEPTH) parametresi kontrol edilir. Satış isteklerinin sayısı aktarılan Piyasa Derinliğindeki bu değeri aşarsa fazlalıklar atılır. Aynısı alış istekleri için de geçerlidir.

'books' dizisinin örnek dolumu:

Piyasa Derinliği durumu	books[] dolumu

	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;

Örnek:

```
//+------------------------------------------------------------------+
//| Uzman Danışman başlatma fonksiyonu                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- veri alacağımız sembol için Piyasa Derinliğini etkinleştir
   MarketBookAdd(Symbol());
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Uzman Danışman sonlandırma fonksiyonu                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
  }
//+------------------------------------------------------------------+
//| Tik fonksiyonu                                                   |
//+------------------------------------------------------------------+
void OnTick(void)
  {
   MqlTick ticks[];
   ArrayResize(ticks,1);
```

```
//--- mevcut fiyatları ortak sembolden özel olana kopyala
   if(SymbolInfoTick(Symbol(),ticks[0]))
     {
      string symbol_name=Symbol()+".SYN";
      CustomTicksAdd(symbol_name,ticks);
     }
  }
//+------------------------------------------------------------------+
//| Book fonksiyonu                                                  |
//+------------------------------------------------------------------+
void OnBookEvent(const string &book_symbol)
  {
//--- mevcut Piyasa Derinliği durumunu ortak sembolden özel olana kopyala
   if(book_symbol==Symbol())
     {
      MqlBookInfo book_array[];
      if(MarketBookGet(Symbol(),book_array))
        {
         string symbol_name=Symbol()+".SYN";
         CustomBookAdd(symbol_name,book_array);
        }
     }
  }
//+------------------------------------------------------------------+
```

Ayrıca bakınız

MarketBookAdd, CustomTicksAdd, OnBookEvent

Çizelge İşlemleri

Grafik özelliklerinin ayarlanması için fonksiyonlar (ChartSetInteger, ChartSetDouble, ChartSetString) eşzamansızdır ve bir tabloya güncelleme komutları göndermek için kullanılır. Bu işlevler başarıyla yürütüldüğünde, komut grafik olaylarının ortak sırasına eklenir. Grafik özellik değişiklikleri, bu grafikteki olayların sırası ile birlikte uygulanır.

Bu nedenle, eşzamanlı olmayan işlevleri çağırdıktan sonra çizelgenin hemen güncellenmesini beklemeyin. Grafik görünümünü ve özelliklerini zorla güncelleştirmek için ChartRedraw() işlevini kullanın.

Fonksiyon	Eylem
ChartApplyTemplate	Belirlenmiş bir dosyadan, belirlenmiş bir şablonu çizelgeye uygular
ChartSaveTemplate	Çizelgenin mevcut ayarlarını belirlenmiş bir isimle bir şablona kaydeder
ChartWindowFind	Göstergelerin çizildiği alt pencerelerin sayısına dönüş yapar
ChartTimePriceToXY	Çizelge koordinatlarını zaman/fiyat ifadesinden, X ve Y koordinatlarına dönüştürür
ChartXYToTimePrice	Bir çizelgedeki X ve Y koordinatlarını, zaman ve fiyat değerlerine dönüştürür
ChartOpen	Belirlenmiş sembol ve periyot ile yeni bir çizelge açar
ChartClose	Belirlenen çizelgeyi kapatır
ChartFirst	Müşteri terminalindeki ilk çizelgenin kimliğine dönüş yapar
ChartNext	Belirlenmiş çizelgenin sonrasındaki, çizelgenin kimliğine dönüş yapar
ChartSymbol	Belirlenen çizelgenin sembol ismine dönüş yapar
ChartPeriod	Belirlenen çizelgenin periyot değerine dönüş yapar
ChartRedraw	Belirlenen çizelgeyi yeniden çizim için zorlar
ChartSetDouble	Belirlenen çizelgenin karşılık gelen özelliği için double değer ayarlar
ChartSetInteger	Belirlenen çizelgenin karşılık gelen özelliği için tamsayı (datetime, int, color, bool or char) değer ayarlar
ChartSetString	Belirlenen çizelgenin karşılık gelen özelliği için string tipli değer ayarlar
ChartGetDouble	Belirlenen çizelgenin double tipli özelliğine dönüş yapar
ChartGetInteger	Belirlenen çizelgenin tamsayı değerli özelliğine dönüş yapar
ChartGetString	Belirlenen çizelgenin string tipli özelliğine dönüş yapar
ChartNavigate	Belirlenen çizelge üzerinde, belirlenen konuma göre, belirlenen değerde kaydırma gerçekleştirir
ChartID	Mevcut çizelgenin kimliğine dönüş yapar

Fonksiyon	Eylem
ChartIndicatorAdd	Belirlenen tanıtıcı değer ile bir göstergeyi, belirlenen bir çizelge penceresine ekler
ChartIndicatorDelete	Belirlenen bir çizelge penceresinden belirlenen isimde bir göstergeyi kaldırır
ChartIndicatorGet	Belirlenen çizelge içinde, belirlenen kısa isimli göstergenin tanıtıcı değerine dönüş yapar
ChartIndicatorName	Belirlenen çizelge üzerindeki gösterge listesindeki numarayı kullanarak göstergenin kısa ismine dönüş yapar
ChartIndicatorsTotal	Belirlenen çizelge penceresine uygulanmış tüm göstergelerin sayısına dönüş yapar
ChartWindowOnDropped	Uzman Danışmanın veya betiğin iliştirildiği çizelge alt penceresinin indisine dönüş yapar
ChartPriceOnDropped	Uzman Danışmanın veya betiğin iliştirildiği çizelge noktasının fiyat koordinatına dönüş yapar
ChartTimeOnDropped	Uzman Danışmanın veya betiğin iliştirildiği çizelge noktasının zaman koordinatına dönüş yapar
ChartXOnDropped	Uzman Danışmanın veya betiğin iliştirildiği çizelge noktasının X koordinatına dönüş yapar
ChartYOnDropped	Uzman Danışmanın veya betiğin iliştirildiği çizelge noktasının Y koordinatına dönüş yapar
ChartSetSymbolPeriod	Belirlenen nesnenin sembol değerini ve periyodunu değiştirir
ChartScreenShot	Çizelgenin mevcut durumunun GIF, PNG veya BMP formatında ekran görüntüsünü alır

ChartApplyTemplate

Belirlenen bir dosyadan, belirli bir şablonu çizelgeye uygular. Komut, çizelge mesajları kuyruğuna eklendi ve önceki komutların işlenmesinin hemen ardından uygulanacak.

```
bool  ChartApplyTemplate(
   long          chart_id,      // Çizelge tanımlayıcı
   const string  filename       // Şablon dosyasının adı
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

filename

[in] Şablonu içeren dosyanın adı.

Dönüş değeri

Komutun çizelge mesajları kuyruğuna eklenmesi durumunda 'true', aksi durumda 'false' dönüşü yapar. Hata hakkında bilgi almak için, GetLastError() fonksiyonunu çağırın.

Not

Yeni bir şablonun başarılı şekilde yüklenmesi durumunda, Uzman Danışman kaldırılacaktır ve işlemine devam edemeyecektir.

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.

Şablonların Kullanılması

MQL5 dili birden çok çizelge özelliğinin ayarlanmasına olanak tanır, bunların arasına ChartSetInteger() fonksiyonu ile renkler de dahildir:

- Çizelge arka plan rengi;
- Eksenlerin, ölçeğin ve OHLC çizgisinin rengi;
- Izgara rengi;
- Hacimlerin ve pozisyon açma seviyelerinin rengi;
- Yukarı yönlü çubuğun, boğa mumunun kenarının ve gölgesinin rengi;
- Aşağı yönlü çubuğun, ayı mumunun kenarının ve gölgesinin rengi;
- Çizelge çizgisinin ve Doji mumlarının rengi;

- Boğa mumunun gövde rengi;
- Ayı mumunun gövde rengi;
- Alış fiyatı çizgisinin rengi;
- Satış fiyatı çizgisinin rengi;
- Son işlem fiyatının (Last) rengi;
- Stop (durdurma) emri seviyesinin (Stop Loss ve Take Profit) rengi.

Çizelgede birden çok grafiksel nesne ve gösterge olabilir. Bir çizelgeyi tüm gerekli göstergelerle bir kez başlatıp, sonra bunları bir şablon olarak saklayabilirsiniz. Böyle bir şablon tüm çizelgelere uygulanabilir.

ChartApplyTemplate() fonksiyonu, daha önce kaydedilmiş bir şablonu kullanmak için düşünülmüştür ve her MQL5 programı içinde kullanılabilir. Şablonun kaydedildiği dosyanın adresi, ChartApplyTemplate() fonksiyonuna ikinci parametre olarak geçirilir. Şablon dosyası şu kurallara göre aranır:

- Eğer ters-bölü "\" ayracı ("\\" olarak yazılır) dosya yolunun başında yer alıyorsa, şablon _terminal_veri_dizini\MQL5 adresine göre aranır,
- Eğer hiç ters-bölü yoksa şablon, içinde ChartApplyTemplate() fonksiyonunun çağrıldığı çalıştırılabilir EX5 dosyasına göre aranır;
- Eğer şablon ilk iki şekilde bulunamazsa, arama terminal_dizini\Profiles\Templates\ klasöründe gerçekleşir.

Burada terminal_dizini, MetaTrader 5 Müşteri Terminalinin çalıştırıldığı konumdur; terminal_veri_dizini ise, düzenlenebilir dosyaların bulunduğu konumdur, bu konum işletim sistemine, kullanıcı ismine ve bilgisayarın güvenlik ayarlarına bağlıdır. Normalde bunlar farklı klasörlerdir ama bazı durumlarda aynı olabilirler.

terminal_veri_dizini ve terminal_dizini klasörlerinin konumu TerminalInfoString() fonksiyonu kullanılarak elde edilebilir.

```
//--- Terminalin başlatıldığı dizin
   string terminal_path=TerminalInfoString(TERMINAL_PATH);
   Print("Terminal dizini:",terminal_path);
//--- içinde Uzman Danışmanların ve göstergelerin yer aldığı MQL5 klasörünün bulunduğu
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
   Print("Terminal veri dizini:",terminal_data_path);
```

Örneğin:

```
//--- şablonu terminal_veri_dizini\MQL5\ konumunda ara
ChartApplyTemplate(0,"\\first_template.tpl"))
//--- şablonu önce EX5_dosyasının_konumu\, sonra terminal_veri_dizini\Profiles\Templat
ChartApplyTemplate(0,"second_template.tpl"))
//--- şablonu önce EX5_dosyasının_konumu\My_templates\, sonra terminal_dizini\Profiles
ChartApplyTemplate(0,"My_templates\\third_template.tpl"))
```

Şablonlar kaynak değildir; çalıştırılabilir EX5 dosyalarına dahil edilemezler.

Örnek:

```
//+------------------------------------------------------------------+
//| 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");
     }
  }
```

Ayrıca Bakınız

Kaynaklar

ChartSaveTemplate

Mevcut çizelge ayarlarını, belirtilen isimle bir şablona kaydeder.

```
bool  ChartSaveTemplate(
   long          chart_id,     // Çizelge tanımlayıcı
   const string  filename      // Şablonu kaydetmek için dosya ismi
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

filename

[in] Şablonun kaydedileceği dosya ismi. ".tpl" uzantısı, dosya ismine otomatik olarak eklenecektir, ayrıyeten belirtilmesine gerek yoktur. Şablon, **terminal_dizini**\Profiles\Templates\ konumuna kaydedilir ve terminaldeki manuel uygulamalar için kullanılabilir. Eğer aynı isimde bir şablon zaten mevcutsa, o zaman bu dosyanın içeriği üzerine yazılacaktır.

Dönüş değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapacaktır. Hata ile ilgili bilgi almak için, GetLastError() fonksiyonunu çağırın.

Not

Şablonları kullanarak, tüm göstergeler ve grafiksel nesnelerle birlikte tüm çizelge ayarlarını saklayabilir; sonra bunları başka bir çizelgeye uygulayabilirsiniz.

Örnek:

```
//+------------------------------------------------------------------+
//|                                      Test_ChartSaveTemplate.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             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
//--- giriş parametreleri
input string               symbol="GBPUSD";  // Yeni çizelgenin sembolü
input ENUM_TIMEFRAMES      period=PERIOD_H3; // Yeni çizelgenin zaman aralığı
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Önce göstergeleri çizelgeye ekle
   int handle;
//--- Göstergeyi kullanım için hazırla
```

```
   if(!PrepareZigzag(NULL,0,handle)) return; // Başarısız, o yüzden çık
//--- Göstergeyi mevcut çizelgeye, ayrı pencerede tuttur.
   if(!ChartIndicatorAdd(0,1,handle))
     {
      PrintFormat("Çizelge %s/%s için %d tanıtıcı değere sahip göstergenin eklenmesi k
                  _Symbol,
                  EnumToString(_Period),
                  handle,
                  GetLastError());
      //--- Program işlemini sonlandır
      return;
     }
//--- Göstergeyi görmek için çizelgeyi yenile
   ChartRedraw();
//--- Zigzagın son iki kırığını bul
   double two_values[];
   datetime two_times[];
   if(!GetLastTwoFractures(two_values,two_times,handle))
     {
      PrintFormat("Zigzagın son iki kırığı bulunamadı!");
      //--- Program işlemini sonlandır
      return;
     }
//--- Şimdi bir standart sapma kanalı ekle
   string channel="StdDeviation Channel";
   if(!ObjectCreate(0,channel,OBJ_STDDEVCHANNEL,0,two_times[1],0))
     {
      PrintFormat("%s nesnesinin oluşturulması başarısız. Hata kodu %d",
                  EnumToString(OBJ_STDDEVCHANNEL),GetLastError());
      return;
     }
   else
     {
      //--- Kanal oluşturuldu, ikinci noktayı tanımla
      ObjectSetInteger(0,channel,OBJPROP_TIME,1,two_times[0]);
      //--- Kanal için bir araç ipucu ayarla
      ObjectSetString(0,channel,OBJPROP_TOOLTIP,"Demo from MQL5 Help");
      //--- Çizelgeyi yenile
      ChartRedraw();
     }
//--- Sonucu bir şablona kaydet
   ChartSaveTemplate(0,"StdDevChannelOnZigzag");
//--- Yeni bir çizelge aç ve kaydedilen şablonu buna uygula
   long new_chart=ChartOpen(symbol,period);
   //--- Araç ipuçlarını, grafiksel nesneler için aktif hale getir
   ChartSetInteger(new_chart,CHART_SHOW_OBJECT_DESCR,true);
   if(new_chart!=0)
     {
      //--- Kaydedilen şablonu bir çizelgeye uygula
```

```
      ChartApplyTemplate(new_chart,"StdDevChannelOnZigzag");
     }
   Sleep(10000);
  }
//+------------------------------------------------------------------+
//| Bir zigzag işleyicisi oluşturur ve verisinin hazır olduğunu temin eder |
//+------------------------------------------------------------------+
bool PrepareZigzag(string sym,ENUM_TIMEFRAMES tf,int &h)
  {
   ResetLastError();
//--- Zigzag göstergesi terminal_veri_klasörü\MQL5\Examples konumuna yerleştirilmiş ol
   h=iCustom(sym,tf,"Examples\\Zigzag");
   if(h==INVALID_HANDLE)
     {
      PrintFormat("%s: Zigzag işleyicisinin oluşturulması başarısız oldu. Hata kodu %d
                  __FUNCTION__,GetLastError());
      return false;
     }
//--- Bir gösterge tanıtıcı değeri oluşturulurken, verilerin hesaplanması zaman alacak
   int k=0; // Gösterge hesabı için bekleme denemelerinin sayısı
//--- Döngü hesabı için bekle, hesaplama hazır değilse 50 milisaniye dur
   while(BarsCalculated(h)<=0)
     {
      k++;
      //--- Deneme sayısını göster
      PrintFormat("%s: k=%d",__FUNCTION__,k);
      //--- Gösterge hesaplanana kadar 50 milisaniye bekle
      Sleep(50);
      //--- yüzden fazla deneme varsa, o halde bir şeyler yanlış
      if(k>100)
        {
         //--- Bir problemi bildir
         PrintFormat("Göstergenin hesaplanması %d denemede başarısız oldu!");
         //--- Program işlemini durdur
         return false;
        }
     }
//--- Her şey hazır, gösterge oluşturuldı ve değerler hesaplandı
   return true;
  }
//+------------------------------------------------------------------+
//| Son iki zigzag kırığını arar ve dizilere yerleştirir             |
//+------------------------------------------------------------------+
bool GetLastTwoFractures(double &get_values[],datetime &get_times[],int handle)
  {
   double values[];          // Zigzag değerleri için bir dizi
   datetime times[];         // Zamanı almak için bir dizi
   int size=100;             // Dizi büyüklüğü
   ResetLastError();
```

```
//--- Göstergenin son 100 değerini kopyala
   int copied=CopyBuffer(handle,0,0,size,values);
//--- Kopyalanan değerlerin sayısını kontrol et
   if(copied<100)
     {
      PrintFormat("%s: %d tanıtıcı değerine sahip göstergenin %d değerinin alınması ba
                  __FUNCTION__,handle,size,GetLastError());
      return false;
     }
//--- Diziye erişimi zaman serilerindeki gibi tanımla
   ArraySetAsSeries(values,true);
//--- Kırılmaların bulunduğu çubukların numaralarını buraya yaz
   int positions[];
//--- Dizi büyüklüğünü ayarla
   ArrayResize(get_values,3); ArrayResize(get_times,3); ArrayResize(positions,3);
//--- Sayaçlar
   int i=0,k=0;
//--- Kırılmaları aramaya başla
   while(i<100)
     {
      double v=values[i];
      //--- Boş değerler ile ilgilenmiyoruz
      if(v!=0.0)
        {
         //--- Çubuk numarasını hatırla
         positions[k]=i;
         //--- Kırılma anındaki zigzag değerini hatırla
         get_values[k]=values[i];
         PrintFormat("%s: Zigzag[%d]=%G",__FUNCTION__,i,values[i]);
         //--- Sayacı artır
         k++;
         //--- Eğer iki kırık bulunduysa döngüyü sonlandır
         if(k>2) break;
        }
      i++;
     }
//--- Dizilere erişimi zaman serilerindeki gibi tanımla
   ArraySetAsSeries(times,true);   ArraySetAsSeries(get_times,true);
   if(CopyTime(_Symbol,_Period,0,size,times)<=0)
     {
      PrintFormat("%s: %d değeri CopyTime()'dan kopyalama başarısız. Hata kodu %d",
                  __FUNCTION__,size,GetLastError());
      return false;
     }
//--- Son iki kırığın gerçekleştiği çubukların açılış zamanını al
   get_times[0]=times[positions[1]];// Ilk kırılma olarak, son değer yazılacak
   get_times[1]=times[positions[2]];// Sondan üçüncü değer ikinci kırılma olacak
   PrintFormat("%s: first=%s,  second=%s",__FUNCTION__,TimeToString(get_times[1]),Time
//--- Başarılı
```

```
   return true;
  }
```

Ayrıca Bakınız

ChartApplyTemplate(), Kaynaklar

ChartWindowFind

Göstergenin eklendiği alt pencerenin numarasına dönüş yapar. Fonksiyonun 2 çeşidi bulunmaktadır.

1. Fonksiyon, belirtilen çizelge içinde göstergenin kısa ismiyle (kısa isim alt pencerenin sol üst köşesinde gözükür) alt pencereyi arar ve başarı durumunda pencerenin numarasına dönüş yapar.

```
int  ChartWindowFind(
   long     chart_id,                  // çizelge tanımlayıcısı
   string   indicator_shortname        // kısa gösterge ismi, bakınız INDICATOR_SHORTN
```

2. Fonksiyon bir özel göstergeden çağrılmalıdır. Göstergenin çalıştığı alt pencerenin numarasına dönüş yapar.

```
int  ChartWindowFind();
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0 mevcut çizelgeyi ifade eder.

indicator_shortname

[in] Göstergenin kısa ismi.

Dönüş değeri

Başarı durumunda, alt pencerenin numarası. Başarısızlık durumunda fonksiyon -1 değerine dönüş yapar.

Not

Eğer fonksiyonun ikinci çeşidi (parametresiz olan) bir scriptten veya, Uzman danışmandan çağrılırsa, fonksiyon -1 değerine dönüş yapar.

iCustom() ve IndicatorCreate() fonksiyonları tarafından gösterge oluşturulduğunda belirlenen dosya ismiyle, gösterge kısa ismini karıştırmayın. Eğer gösterge kısa ismi açık şekilde belirtilmezse, gösterge kaynak kodunu içeren dosyanın ismi, derleme sırasında kısa isim olarak belirlenir.

INDICATOR_SHORTNAME özelliğinde IndicatorSetString() fonksiyonu ile kaydedilen gösterge kısa ismini doğru şekillendirmek önemlidir. Kısa ismin, gösterge giriş parametrelerini içermesi önerilir, göstergenin çizelgeden kaldırılması için kullanılan ChartIndicatorDelete() fonksiyonu kısa isimle tanımlanır.

Örnek:

```
#property script_show_inputs
//--- giriş parametreleri
input string   shortname="MACD(12,26,9)";
//+------------------------------------------------------------------+
//| Göstergenin bulunduğu çizelge penceresinin numarasına dönüş yapar  |
//+------------------------------------------------------------------+
int GetIndicatorSubWindowNumber(long chartID=0,string short_name="")
  {
   int window=-1;
```

```
//---
   if((ENUM_PROGRAM_TYPE)MQL5InfoInteger(MQL5_PROGRAM_TYPE)==PROGRAM_INDICATOR)
     {
      //--- fonksiyon göstergeden çağrıldı, isim gerekmiyor.
      window=ChartWindowFind();
     }
   else
     {
      //--- fonksiyon bir Uzman Danışmandan veya scriptten çağrılır
      window=ChartWindowFind(0,short_name);
      if(window==-1) Print(__FUNCTION__+"(): Error = ",GetLastError());
     }
//---
   return(window);
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//---
   int window=GetIndicatorSubWindowNumber(0,shortname);
   if(window!=-1)
      Print(+shortname+" göstergesi #"+(string)window)+" penceresindedir");
   else
      Print(shortname+" göstergesi şurada bulunamadı pencere = "+(string)window);
  }
```

Ayrıca Bakınız

ObjectCreate(), ObjectFind()

ChartTimePriceToXY

Çizelge koordinatlarını zaman/fiyat ifadesinden X ve Y ifadesine dönüştürür.

```
bool  ChartTimePriceToXY(
   long           chart_id,      // Çizelge tanımlayıcı
   int            sub_window,    // Alt pencerenin numarası
   datetime       time,          // Çizelge üstündeki zaman
   double         price,         // Çizelge üstündeki fiyat
   int&           x,             // Çizelge üstündeki zaman için X koordinatı
   int&           y              // Çizelge üstündeki fiyat için Y koordinatı
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

sub_window

[in] Çizelge alt penceresinin numarası. 0, ana çizelge penceresi demektir.

time

[in] Çizelgedeki zaman değeri, değer X ekseninde pikseller olarak alınacaktır. Orjin, ana çizelge penceresi sol üst köşesidir.

price

[in] Çizelgedeki fiyat değeri, değer Y ekseninde pikseller olarak alınacaktır. Orjin, ana çizelge penceresi sol üst köşesidir.

x

[out] Zaman ekseninden X eksenine dönüşümün yapılacağı değişken.

y

[out] Fiyat ekseninden Y eksenine dönüşümün yapılacağı değişken.

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata ile ilgili bilgi almak için, GetLastError() fonksiyonunu çağırın.

Ayrıca Bakınız

ChartXYToTimePrice()

ChartXYToTimePrice

Çizelge üzerindeki X ve Y koordinatlarını zaman ve fiyat değerlerine çevirir.

```
bool  ChartXYToTimePrice(
   long            chart_id,      // Çizelge tanımlayıcı
   int             x,             // Çizelgedeki X koordinatı
   int             y,             // Çizelgedeki Y koordinatı
   int&            sub_window,    // Alt pencere numarası
   datetime&       time,          // Çizelgedeki zaman
   double&         price          // Çizelgedeki fiyat
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

x

[in] X koordinatı.

y

[in] Y koordinatı.

sub_window

[out] Alt pencere numarasının yazıldığı değişken. 0, ana çizelge penceresi demektir.

time

[out] X ekseni boyunca piksellerin alınacağı, çizelge üzerindeki zaman değeri. Orjin, ana çizelge pencerenin sol üst köşesidir.

price

[out] Y ekseni boyunca piksellerin alınacağı, çizelge üzerindeki fiyat değeri. Orjin, ana çizelge pencerenin sol üst köşesidir.

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata ile ilgili bilgi almak için, GetLastError() fonksiyonunu çağırın.

Örnek:

```
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- Olay parametrelerini çizelge üzerinde göster
   Comment(__FUNCTION__,": id=",id," lparam=",lparam," dparam=",dparam," sparam=",spa
```

```
//--- Eğer bu, çizelge üzerinde bir fare tıklama olayıysa
   if(id==CHARTEVENT_CLICK)
     {
      //--- Değişkenleri ayarla
      int      x     =(int)lparam;
      int      y     =(int)dparam;
      datetime dt    =0;
      double   price =0;
      int      window=0;
      //--- X ve Y koordinatlarını zaman/fiyat bağlamına dönüştür
      if(ChartXYToTimePrice(0,x,y,window,dt,price))
        {
         PrintFormat("Window=%d X=%d  Y=%d  =>  Time=%s  Price=%G",window,x,y,TimeToSt
         //--- Ters dönüşüm gerçekleştir: (X,Y) => (Zaman,Fiyat)
         if(ChartTimePriceToXY(0,window,dt,price,x,y))
            PrintFormat("Time=%s  Price=%G  =>  X=%d  Y=%d",TimeToString(dt),price,x,y
         else
            Print("ChartTimePriceToXY şu hata koduna dön: ",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 hata kodunu dön: ",GetLastError());
      Print("+------------------------------------------------------------------+");
     }
  }
```

Ayrıca Bakınız

ChartTimePriceToXY()

ChartOpen

Belirtilen sembol ve periyot ile yeni bir çizelge açar.

```
long  ChartOpen(
   string           symbol,     // Sembol ismi
   ENUM_TIMEFRAMES  period      // Periyot
   );
```

Parametreler

symbol

[in] Çizelge sembolü. NULL değeri mevcut (Uzman Danışmanın iliştirildiği) çizelge sembolü anlamına gelir.

period

[in] Çizelge periyodu (zaman aralığı). ENUM_TIMEFRAMES değerlerinden biri olabilir. 0 mevcut çizelge periyodu anlamına gelir.

Dönüş değeri

Başarılı ise, açılan çizelgenin kimliğine dönüş yapar. Aksi durumda 0 değerine dönüş yapar.

Not

Terminalde açılan çizelgelerin sayısı CHARTS_MAX değerini geçemez.

ChartFirst

Müşteri terminalindeki ilk grafiğin kimliğine dönüş yapar.

```
long  ChartFirst();
```

Dönüş değeri

Çizelge tanımlayıcı.

ChartNext

Belirtilmiş olanın ardındaki çizelgenin kimliğine dönüş yapar.

```
long  ChartNext(
   long  chart_id      // Çizelge tanımlayıcı
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0 mevcut çizelgeyi belirtmez. 0 "ilk çizelge kimliğine dön" anlamındadır.

Dönüş değeri

Çizelge kimliği. Eğer çizelge listesinin sonunda bulunuluyorsa, -1 değerine dönüş yapar.

Örnek:

```
//--- çizelge kimliği için değişkenler
   long currChart,prevChart=ChartFirst();
   int i=0,limit=100;
   Print("ChartFirst =",ChartSymbol(prevChart)," ID =",prevChart);
   while(i<limit)// 100'den fazla çizelgemiz olmadığı kesin
     {
      currChart=ChartNext(prevChart); // Önceki çizelgenin kimliğini kullanarak, yeni
      if(currChart<0) break;          // Çizelge listesinin sonuna ulaşıldı
      Print(i,ChartSymbol(currChart)," ID =",currChart);
      prevChart=currChart;// ChartNext() için mevcut çizelge kimliğini kaydet
      i++;// Sayacı artırmayı unutma
     }
```

ChartClose

Belirtilen çizelgeyi kapatır.

```
bool  ChartClose(
   long  chart_id=0      // Çizelge tanımlayıcı
   );
```

Parametreler

chart_id=0

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

Dönüş değeri

Başarılı sonuç durumunda 'true', aksi durumda 'false' dönüşü yapar.

ChartSymbol

Belirtilen grafik için sembol ismine dönüş yapar.

```
string  ChartSymbol(
   long  chart_id=0      // Çizelge tanımlayıcı
   );
```

Parametreler

chart_id=0

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

Dönüş değeri

Eğer çizelge mevcut değilse, sonuç boş bir dizgidir.

Ayrıca Bakınız

ChartSetSymbolPeriod

ChartPeriod

Belirtilmiş çizelgenin zaman dilimi periyoduna dönüş yapar.

```
ENUM_TIMEFRAMES  ChartPeriod(
   long  chart_id=0      // Çizelge tanımlayıcı
   );
```

Parametreler

chart_id=0

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

Dönüş değeri

Fonksiyon ENUM_TIMEFRAMES değerlerinden birine dönüş yapar. Eğer çizelge mevcut değilse, 0'a döner .

ChartRedraw

Bu fonksiyon, belirtilen çizelgeyi yeniden çizime zorlar.

```
void  ChartRedraw(
   long  chart_id=0      // Çizelge tanımlayıcı
   );
```

Parametreler

chart_id=0

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

Not

Bu genellikle nesne özelliklerinden sonra kullanılır.

Ayrıca Bakınız

Nesne fonksiyonları

ChartSetDouble

Belirlenen çizelgenin karşılık gelen özelliği için değer ayarlar. Çizelge özelliği double tipinde olmalıdır. Komut, çizelge mesajları kuyruğuna eklendi ve önceki komutların işlenmesinin hemen ardından uygulanacak.

```
bool  ChartSetDouble(
   long                           chart_id,      // Çizelge tanımlayıcısı
   ENUM_CHART_PROPERTY_DOUBLE     prop_id,       // Özellik tanımlayıcısı
   double                         value          // Değer
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

prop_id

[in] Çizelge özelliğinin tanımlayıcısı. ENUM_CHART_PROPERTY_DOUBLE değerlerinden biri olabilir (sadece-okunur olanlar hariç).

value

[in] Özellik değeri.

Dönüş değeri

Komutun çizelge mesajları kuyruğuna eklenmesi durumunda 'true', aksi durumda 'false' dönüşü yapar. Hata hakkında bilgi almak için, GetLastError() fonksiyonunu çağırın.

Not

İşlev eşzamansızdır; bu, işlevin, belirtilen grafik sırasına başarıyla eklenmiş olan komutun yürütülmesini beklemediği anlamına gelir. Bunun yerine, kontrolü hemen geri döndürür. Özellik yalnızca, uygun komutun grafik sırasından kaldırılmasından sonra değişecektir. Grafik sırasından komutları hemen yürütmek için ChartRedraw işlevini çağırın.

Bir kerede birkaç grafik özelliğini hemen değiştirmek isterseniz, o zaman ilgili işlevler (ChartSetString, ChartSetDouble, ChartSetString) bir kod bloğunda çalıştırılmalı, sonra aramanız gerekir ChartRedraw bir kere.

Komut yürütme sonucunu kontrol etmek için, belirtilen grafik özelliğini isteyen bir işlev kullanabilirsiniz. (ChartGetInteger, ChartGetDouble, ChartSetString). Ancak, bu işlevlerin senkron olduğunu ve yürütme sonuçlarını beklediğini unutmayın.

ChartSetInteger

Belirlenen çizelgenin karşılık gelen özelliği için değer ayarlar. Çizelge özeliği datetime, int, color, bool veya char tiplerinden biri olmalıdır. Komut, çizelge mesajları kuyruğuna eklendi ve önceki komutların işlenmesinin hemen ardından uygulanacak.

```
bool  ChartSetInteger(
   long                           chart_id,      // Çizelge tanımlayıcı
   ENUM_CHART_PROPERTY_INTEGER    prop_id,       // Özellik tanımlayıcı
   long                           value          // Değer
   );
```

```
bool  ChartSetInteger(
   long                           chart_id,      // Çizelge tanımlayıcı
   ENUM_CHART_PROPERTY_INTEGER    prop_id,       // Özellik tanımlayıcı
   int                            sub_window,    // Alt pencere indisi
   long                           value          // Değer
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

prop_id

[in] Çizelge özelliğinin tanımlayıcısı. ENUM_CHART_PROPERTY_INTEGER değerlerinden biri olabilir (sadece okunur olanlar hariç).

sub_window

[in] Çizelge alt penceresinin indisi. İlk durumda varsayılan değer 0'dır (ana çizelge penceresi). Çoğu özellik alt pencere numarasını gerektirmez.

value

[in] Özellik değeri.

Dönüş değeri

Komutun çizelge mesajları kuyruğuna eklenmesi durumunda 'true', aksi durumda 'false' dönüşü yapar. Hata hakkında bilgi almak için, GetLastError() fonksiyonunu çağırın.

Not

İşlev eşzamansızdır; bu, işlevin, belirtilen grafik sırasına başarıyla eklenmiş olan komutun yürütülmesini beklemediği anlamına gelir. Bunun yerine, kontrolü hemen geri döndürür. Özellik yalnızca, uygun komutun grafik sırasından kaldırılmasından sonra değişecektir. Grafik sırasından komutları hemen yürütmek için ChartRedraw işlevini çağırın.

Bir kerede birkaç grafik özelliğini hemen değiştirmek isterseniz, o zaman ilgili işlevler (ChartSetString, ChartSetDouble, ChartSetString) bir kod bloğunda çalıştırılmalı, sonra aramanız gerekir ChartRedraw bir kere.

Komut yürütme sonucunu kontrol etmek için, belirtilen grafik özelliğini isteyen bir işlev kullanabilirsiniz. (ChartGetInteger, ChartGetDouble, ChartSetString). Ancak, bu işlevlerin senkron olduğunu ve yürütme sonuçlarını beklediğini unutmayın.

Örneğin:

```
//+------------------------------------------------------------------+
//| Uzman başlatma fonksiyonu                                        |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- Grafik penceresinde fare hareketlerini etkinleştirme
   ChartSetInteger(0,CHART_EVENT_MOUSE_MOVE,1);
//--- Grafik özelliklerinin zorla güncellenmesi, olay işleme için hazırlık sağlar
   ChartRedraw();
  }
//+------------------------------------------------------------------+
//| Fare Durumu                                                      |
//+------------------------------------------------------------------+
string MouseState(uint state)
  {
   string res;
   res+="\nML: "   +(((state& 1)== 1)?"DN":"UP");   // fare sol
   res+="\nMR: "   +(((state& 2)== 2)?"DN":"UP");   // fare sağ
   res+="\nMM: "   +(((state&16)==16)?"DN":"UP");   // fare orta
   res+="\nMX: "   +(((state&32)==32)?"DN":"UP");   // fare ilk X tuşu
   res+="\nMY: "   +(((state&64)==64)?"DN":"UP");   // fare ikinci X tuşu
   res+="\nSHIFT: "+(((state& 4)== 4)?"DN":"UP");   // shift tuşu
   res+="\nCTRL: " +(((state& 8)== 8)?"DN":"UP");   // control tuşu
   return(res);
  }
//+------------------------------------------------------------------+
//| ChartEvent fonksiyonu                                            |
//+------------------------------------------------------------------+
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

Belirlenen çizelgenin karşılık gelen özelliği için değer ayarlar. Çizelge özelliği sring tipinde olmalıdır. Komut, çizelge mesajları kuyruğuna eklendi ve önceki komutların işlenmesinin hemen ardından uygulanacak.

```
bool  ChartSetString(
   long                          chart_id,      // Çizelge tanımlayıcı
   ENUM_CHART_PROPERTY_STRING    prop_id,       // Özellik tanımlayıcı
   string                        str_value      // Değer
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

prop_id

[in] Çizelge özelliğinin tanımlayıcısı. Değeri, ENUM_CHART_PROPERTY_STRING değerlerinden biri olabilir (sadece-okunur olanlar hariç).

str_value

[in] Özellik değeri dizgisi. Dizgi uzunluğu 2045 karakteri geçemez (fazla karakterler budanacaktır).

Dönüş değeri

Komutun çizelge mesajları kuyruğuna eklenmesi durumunda 'true', aksi durumda 'false' dönüşü yapar. Hata hakkında bilgi almak için, GetLastError() fonksiyonunu çağırın.

Not

ChartSetString, çizelge üstündeki yorum çıktıları için, Comment fonksiyonunun yerine kullanılabilir.

İşlev eşzamansızdır; bu, işlevin, belirtilen grafik sırasına başarıyla eklenmiş olan komutun yürütülmesini beklemediği anlamına gelir. Bunun yerine, kontrolü hemen geri döndürür. Özellik yalnızca, uygun komutun grafik sırasından kaldırılmasından sonra değişecektir. Grafik sırasından komutları hemen yürütmek için ChartRedraw işlevini çağırın.

Bir kerede birkaç grafik özelliğini hemen değiştirmek isterseniz, o zaman ilgili işlevler (ChartSetString, ChartSetDouble, ChartSetString) bir kod bloğunda çalıştırılmalı, sonra aramanız gerekir ChartRedraw bir kere.

Komut yürütme sonucunu kontrol etmek için, belirtilen grafik özelliğini isteyen bir işlev kullanabilirsiniz. (ChartGetInteger, ChartGetDouble, ChartSetString). Ancak, bu işlevlerin senkron olduğunu ve yürütme sonuçlarını beklediğini unutmayın.

Örnek:

```
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);
  }
```

Ayrıca Bakınız

Comment, ChartGetString

ChartGetDouble

Belirlenen çizelgenin karşılık gelen özellik değerine dönüş yapar. Çizelge özelliği double tipinde olmalıdır. Fonksiyon çağrıları için iki tür bulunmaktadır.

1. Doğrudan özellik değerine dönüş yapar.

```
double  ChartGetDouble(
   long                          chart_id,         // Çizelge tanımlayıcı
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,          // Özellik tanımlayıcı
   int                           sub_window=0      // Eğer gerekliyse, alt pencere indis
   );
```

2. Fonksiyonun başarı durumuna göre, 'true' veya 'false' değerine dönüş yapar. Başarı durumunda özellik değeri, referansla geçirilen bir hedef değişkeni double_var içine yerleştirilir.

```
bool  ChartGetDouble(
   long                          chart_id,         // Çizelge tanımlayıcı
   ENUM_CHART_PROPERTY_DOUBLE    prop_id,          // Özellik tanımlayıcı
   int                           sub_window,       // Alt pencere indisi
   double&                       double_var        // Çizelge özelliği için hedef değiş
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

prop_id

[in] Çizelge özelliğinin tanımlayıcısı. Bu değer, ENUM_CHART_PROPERTY_DOUBLE sayımının değerlerinden biri olabilir.

sub_window

[in] Çizelge alt penceresinin indisi. İlk durumda varsayılan değer 0'dır (ana çizelge penceresi). Çoğu özellik alt pencere numarasını gerektirmez.

double_var

[out] İstenen özellik için double tipli hedef değişkeni.

Dönüş değeri

double tipli değer.

İkinci durumda, eğer özellik mevcutsa ve double_var değişkenine yerleştirilmişse 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata ile ilgili daha fazla bilgi almak için, GetLastError() fonksiyonunun çağrılması gerekir.

Not

İşlev eşzamanlıdır, başka bir deyişle, aramadan önce grafik sırasına eklenen tüm komutların yürütülmesini bekler.

Örnek:

```
void OnStart()
  {
   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

Belirlenen çizelgenin karşılık gelen özellik değerine dönüş yapar. Çizelge özelliği datetime, int veya bool tiplerinden biri olmalıdır. Fonksiyon çağrıları için iki tür bulunmaktadır.

1. Doğrudan özellik değerine dönüş yapar.

```
long  ChartGetInteger(
   long                          chart_id,         // Çizelge tanımlayıcı
   ENUM_CHART_PROPERTY_INTEGER   prop_id,          // Özellik tanımlayıcı
   int                           sub_window=0      // Eğer gerekliyse, alt pencere indi
   );
```

2. Fonksiyonun başarı durumuna göre, 'true' veya 'false' değerine dönüş yapar. Başarı durumunda özellik değeri, referansla geçirilen bir hedef değişkeni long_var içine yerleştirilir.

```
bool  ChartGetInteger(
   long                          chart_id,         // Çizelge tanımlayıcı
   ENUM_CHART_PROPERTY_INTEGER   prop_id,          // Özellik tanımlayıcı
   int                           sub_window,       // Alt pencere indisi
   long&                         long_var          // Özellik için hedef değişken
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

prop_id

[in] Çizelge özelliğinin tanımlayıcısı. Bu değer, ENUM_CHART_PROPERTY_INTEGER sayımının değerlerinden biri olabilir.

sub_window

[in] Çizelge alt penceresinin indisi. İlk durumda varsayılan değer 0'dır (ana çizelge penceresi). Çoğu özellik alt pencere numarasını gerektirmez.

long_var

[out] İstenen özellik için long tipli hedef değişken.

Dönüş değeri

long tipli değer.

İkinci durumda, eğer özellik mevcutsa ve long_var değişkenine yerleştirilmişse 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata ile ilgili daha fazla bilgi için, GetLastError() fonksiyonunun çağrılması gerekmektedir.

Not

İşlev eşzamanlıdır, başka bir deyişle, aramadan önce grafik sırasına eklenen tüm komutların yürütülmesini bekler.

Örnek:

```
void OnStart()
  {
   int height=ChartGetInteger(0,CHART_HEIGHT_IN_PIXELS,0);
   int width=ChartGetInteger(0,CHART_WIDTH_IN_PIXELS,0);
   Print("CHART_HEIGHT_IN_PIXELS =",height,"piksel");
   Print("CHART_WIDTH_IN_PIXELS =",width,"piksel");
  }
```

ChartGetString

Belirlenen çizelgenin karşılık gelen özellik değerine dönüş yapar. Çizelge özelliği string tipinde olmalıdır. Fonksiyon çağrısının iki biçimi bulunmaktadır.

1. Doğrudan özellik değerine dönüş yapar.

```
string  ChartGetString(
   long                        chart_id,         // Çizelge tanımlayıcı
   ENUM_CHART_PROPERTY_STRING  prop_id           // Özellik tanımlayıcı
   );
```

2. Fonksiyonun başarı durumuna göre, 'true' veya 'false' değerine dönüş yapar. Başarı durumunda özellik değeri, referansla geçirilen bir hedef değişkeni string_var içine yerleştirilir.

```
bool  ChartGetString(
   long                        chart_id,         // Çizelge tanımlayıcı
   ENUM_CHART_PROPERTY_STRING  prop_id,          // Özellik tanımlayıcı
   string&                     string_var        // Özellik için hedef değişkeni
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

prop_id

[in] Çizelge özelliğinin tanımlayıcısı. Bu değer ENUM_CHART_PROPERTY_STRING değerlerinden biri olabilir.

string_var

[out] İstenen özellik için string tipli hedef değişkeni.

Dönüş değeri

string tipli değer.

İkinci durumda, eğer özellik mevcutsa ve string_var değişkenine yerleştirilmişse 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata ile ilgili daha fazla bilgi için, GetLastError() fonksiyonunun çağrılması gerekmektedir.

Not

Comment veya ChartSetString fonksiyonları kullanılarak çizelge üzerinde sergilenmiş yorumları okumak için, ChartGetString fonksiyonu kullanılabilir.

İşlev eşzamanlıdır, başka bir deyişle, aramadan önce grafik sırasına eklenen tüm komutların yürütülmesini bekler.

Örnek:

```
void OnStart()
  {
   ChartSetString(0,CHART_COMMENT,"Test comment.\nSecond line.\nThird!");
```

```
   ChartRedraw();
   Sleep(1000);
   string comm=ChartGetString(0,CHART_COMMENT);
   Print(comm);
  }
```

Ayrıca Bakınız

Comment, ChartSetString

ChartNavigate

Çizelgede belirlenen konuma bağlı olarak, belirtilen çizelgenin, belirlenen çubuk sayısı kadar kaydırılmasını sağlar.

```
bool  ChartNavigate(
   long                  chart_id,      // Çizelge tanımlayıcı
   ENUM_CHART_POSITION   position,      // Konum
   int                   shift=0        // Kaydırma değeri
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

position

[in] Kaydırmanın gerçekleştirileceği çizelge konumu. ENUM_CHART_POSITION değerlerinden biri olabilir.

shift=0

[in] Çizelgenin kaydırılacağı çubuk sayısı. Pozitif sayılar, sağa kaydırma anlamına gelir (çizelgenin sonuna doğru); negatif sayılar ise sola kaydırma anlamına gelir (çizelgenin başına doğru). Sıfır kaydırma değeri çizelgenin başına veya sonuna gidilmek için kullanılabilir.

Dönüş değeri

Başarı durumunda 'true', aksi durumda 'false' dönüşü yapar.

Örnek:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- mevcut çizelgenin işleyicisini al
   long handle=ChartID();
   string comm="";
   if(handle>0) // başarılıysa, çizelgeyi ayrıyeten başlat
     {
      //--- otomatik kaydırmayı devre dışı bırak
      ChartSetInteger(handle,CHART_AUTOSCROLL,false);
      //--- sağ çizelge sınırından bir kaydırma ayarla
      ChartSetInteger(handle,CHART_SHIFT,true);
      //--- mumları çiz
      ChartSetInteger(handle,CHART_MODE,CHART_CANDLES);
      //--- görünüm modunu tik hacimleri için ayarla
      ChartSetInteger(handle,CHART_SHOW_VOLUMES,CHART_VOLUME_TICK);

      //--- Comment() çıktısı için bir metin ayarla
      comm="Tarih başlangıcının sağına doğru 10 çubuk kaydır";
```

```
   //--- yorumu göster
   Comment(comm);
   //--- tarih başlangıcının sağına doğru 10 çubuk kaydır
   ChartNavigate(handle,CHART_BEGIN,10);
   //--- çizelgede görünen ilk çubuğun indisini al (indisleme zaman serilerindeki ç
   long first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
   //--- satır atlama karakterini ekle
   comm=comm+"\r\n";
   //--- yoruma ekle
   comm=comm+"Çizelgedeki ilk çubuğun indisi "+IntegerToString(first_bar)+"\r\n";
   //--- yorumu göster
   Comment(comm);
   //--- çizelgenin nasıl hareket edeceğini görmek için 5 saniye bekle
   Sleep(5000);

   //--- yorum metnine ekle
   comm=comm+"\r\n"+"Sağ çizelge sınırına doğru 10 çubuk kaydır";
   Comment(comm);
   //--- sağ çizelge sınırının soluna doğru 10 çubuk kaydır
   ChartNavigate(handle,CHART_END,-10);
   //--- çizelgede görünen ilk çubuğun indisini al (indisleme zaman serilerindeki ç
   first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
   comm=comm+"\r\n";
   comm=comm+"Çizelgedeki ilk çubuğun indisi "+IntegerToString(first_bar)+"\r\n";
   Comment(comm);
   //--- çizelgenin nasıl hareket edeceğini görmek için 5 saniye bekle
   Sleep(5000);

   //--- yeni çizelge kaydırma bloğu
   comm=comm+"\r\n"+"Tarih başlangıcına doğru 300 çubuk sağa kaydır";
   Comment(comm);
   //--- tarih başlangıcına doğru 300 çubuk sağa kaydır
   ChartNavigate(handle,CHART_BEGIN,300);
   first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
   comm=comm+"\r\n";
   comm=comm+"Çizelgedeki ilk çubuğun indisi "+IntegerToString(first_bar)+"\r\n";
   Comment(comm);
   //--- çizelgenin nasıl hareket edeceğini görmek için 5 saniye bekle
   Sleep(5000);

   //--- yeni çizelge kaydırma bloğu
   comm=comm+"\r\n"+"Sağ çizelge sınırının soluna doğru 300 çubuk kaydır";
   Comment(comm);
   //--- sağ çizelge sınırının soluna doğru 300 çubuk kaydır
   ChartNavigate(handle,CHART_END,-300);
   first_bar=ChartGetInteger(0,CHART_FIRST_VISIBLE_BAR,0);
   comm=comm+"\r\n";
   comm=comm+"Çizelgedeki ilk çubuğun indisi "+IntegerToString(first_bar)+"\r\n";
   Comment(comm);
```

```
      }
   }
```

ChartID

Mevcut çizelgenin kimliğine dönüş yapar.

```
long  ChartID();
```

Dönüş değeri

long tipli değer.

ChartIndicatorAdd

Belirlenmiş tanıtıcı değere sahip olan bir göstergeyi, belirlenen bir çizelgeye ekler. Gösterge ve çizelge aynı sembol ve aynı zaman aralığı için oluşturulmuş olmalıdır.

```
bool  ChartIndicatorAdd(
   long  chart_id,                  // çizelge tanımlayıcı
   int   sub_window                 // alt pencere numarası
   int   indicator_handle           // gösterge tanıtıcı değeri
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

sub_window

[in] çizelge alt penceresinin numarası. 0, ana çizelge penceresi demektir. Yeni bir pencerede bir gösterge eklemek için parametre, mevcut son pencerenin numarasından bir büyük olmalıdır; yani CHART_WINDOWS_TOTAL değerine eşit olmalıdır. Eğer parametre değeri, CHART_WINDOWS_TOTAL değerinden büyükse, yeni pencere oluşturulmayacak ve gösterge eklenmeyecektir.

indicator_handle

[in] Göstergenin tanıtıcı değeri.

Dönüş değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapacaktır. Hata ile ilgili bilgi almak için, GetLastError() fonksiyonunu çağırın. Hata 4114, çizelgenin ve göstergenin, zaman aralığı veya sembol açısından farklı olduklarını belirtir.

Not

Eğer ayrı bir alt pencerede çizilmesi gereken bir gösterge (örneğin, gömülü iMACD veya #property indicator_separate_window özelliğiyle belirlenmiş bir özel gösterge), ana pencereye uygulanmışsa, bu pencerede görünür olmayabilir ama göstergeler listesinde yer alır. Bu, gösterge ölçeğinin fiyat çizelgesi ölçeğinden farklı olduğu ve uygulanan göstergenin değerlerinin, fiyat çizelgesinin kapsamına uymadığı anlamına gelir. Bu durumda GetLastError()fonksiyonu, hata olmadığını belirten sıfır değerine dönüş yapar. Bunun gibi "görünmez" göstergeler, Veri Penceresinde görülebilir ve diğer MQL5 uygulamaları tarafından kullanılabilir.

Örnek:

```
#property description "ChartIndicatorAdd() fonksiyonu ile çalışmayı gösteren Uzman Dan
#property description "Grafik üzerinde çalıştırdıktan (ve hata mesajı aldıktan) sonra,
#property description "özelliklerini açın ve doğru <symbol> ve <period> parametrelerin
#property description "MACD göstergesi çizelgeye eklenecektir."

//--- giriş parametreleri
input string          symbol="AUDUSD";     // sembol ismi
input ENUM_TIMEFRAMES period=PERIOD_M12;   // zaman aralığı
input int     fast_ema_period=12;          // hızlı MACD periyodu
```

```
input int    slow_ema_period=26;          // yavaş MACD periyodu
input int      signal_period=9;           // sinyal periyodu
input ENUM_APPLIED_PRICE apr=PRICE_CLOSE; // MACD hesabı için fiyat tipi

int indicator_handle=INVALID_HANDLE;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//---
   indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_period,
//--- göstergeyi çizelgeye eklemeyi dene
   if(!AddIndicator())
     {
      //--- AddIndicator() fonksiyonu, göstergeyi çizelgeye eklemeyi reddeti
      int answer=MessageBox("MACD'yi hala çizelgeye eklemek istiyor musunuz?",
                            "göstergeyi eklemek için hatalı sembol ve/veya zaman aral
                            MB_YESNO // "Evet" ve "Hayır" seçim düğmeleri gözükecek
                            );
      //--- kullanıcı hala ChartIndicatorAdd() fonksiyonunun hatalı kullanımında ısra
      if(answer==IDYES)
        {
         //--- öncelikle, bunun hakkında bir bülten girişi yapılacak
         PrintFormat("Uyarı! %s: MACD(%s/%s) göstergesi %s/%s çizelgesine eklenmek ist
                     __FUNCTION__,symbol,EnumToString(period),_Symbol,EnumToString(_Pe
         //--- göstergeyi eklemeyi deneyeceğimiz, yeni bir alt pencere numarası al
         int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
         //--- şimdi başarısızlığa mahkum bir deneme yap
         if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
            PrintFormat("MACD göstergesini %d çizelge penceresine ekleme başarısız olu
                        subwindow,GetLastError());
        }
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
// Uzman Danışman hiçbir şey gerçekleştirmiyor
  }
//+------------------------------------------------------------------+
//| Göstergeyi kontrol edip çizelgeye eklemek için fonksiyon         |
//+------------------------------------------------------------------+
bool AddIndicator()
  {
```

```
//--- gösterilen mesaj
   string message;
//--- gösterge ve çizelge sembollerinin uyumunu kontrol et
   if(symbol!=_Symbol)
     {
      message="Demo_ChartIndicatorAdd() fonksiyonunun kullanımını görüntüle:";
      message=message+"\r\n";
      message=message+"Başka sembolde hesaplanan göstergenin çizelgeye eklenmesi başa
      message=message+"\r\n";
      message=message+"Uzman Danışman özelliğinde çizelge sembolünü belirt - "+_Symbol
      Alert(message);
      //--- erken çıkış, gösterge çizelgeye eklenmeyecek
      return false;
     }
//--- göstergenin ve çizelgenin zaman aralıkları uyuşuyor mu kontrol et
   if(period!=_Period)
     {
      message="farklı zaman aralığında hesaplanmış gösterge, çizelgeye eklemiyor.";
      message=message+"\r\n";
      message=message+"Uzman Danışman özelliklerinde çiizelge zaman aralığını belirt -
      Alert(message);
      //--- erken çıkış, gösterge çizelgeye eklenmeyecek
      return false;
     }
//--- tüm kontroller tamamlandı, sembol ve gösterge zaman aralığı çizelgeyle uygun
   if(indicator_handle==INVALID_HANDLE)
     {
      Print(__FUNCTION__,"  MACD göstergesi oluşturuluyor");
      indicator_handle=iMACD(symbol,period,fast_ema_period,slow_ema_period,signal_peri
      if(indicator_handle==INVALID_HANDLE)
        {
         Print("MACD göstergesinin oluşturulması başarısız oldu. Hata kodu ",GetLastEr
        }
     }
//--- hata kodunu sıfırla
   ResetLastError();
//--- göstergeyi çizelgeye uygula
   Print(__FUNCTION__,"  MACD göstergesinin çizelgeye eklenmesi");
   Print("MACD şu değerlerle oluşturuldu ",symbol,"/",EnumToString(period));
//--- MACD göstergesinin eklendiği alt pencere numarasını al
   int subwindow=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   PrintFormat("MACD göstergesinin %d çizelge penceresine eklenmesi",subwindow);
   if(!ChartIndicatorAdd(0,subwindow,indicator_handle))
     {
      PrintFormat("MACD göstergesinin %d çizelge penceresine eklenmesi başarısız oldu
                  subwindow,GetLastError());
     }
//--- Gösterge başarıyla eklendi
   return(true);
```

```
}
```

Ayrıca Bakınız

ChartIndicatorDelete(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate()

ChartIndicatorDelete

Belirtilen çizelgeden belirtilen bir göstergeyi kaldırır.

```
bool  ChartIndicatorDelete(
   long           chart_id,              // çizelge tanımlayıcı
   int            sub_window             // alt pencere numarası
   const string   indicator_shortname    // göstergenin kısa ismi
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0 mevcut çizelgeyi ifade eder.

sub_window

[in] Çizelge alt penceresinin indisi. 0 ana çizelge penceresini tanımlar.

const indicator_shortname

[in] INDICATOR_SHORTNAME özelliğinde IndicatorSetString() fonksiyonu ile ayarlanan, gösterge kısa ismi. Gösterge kısa ismini öğrenmek için ChartIndicatorName() fonksiyonunu kullanın.

Dönüş değeri

Göstergenin silinmesinin başarılı olması durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata detayları için GetLastError() fonksiyonunu kullanın.

Not

Eğer çizelge penceresinde aynı isimli birden fazla gösterge varsa, ilki silinecektir.

Eğer başka göstergeler bu silinen göstergenin verilerine dayanıyorsa, onlar da silinecektir.

Gösterge oluşturulurken iCustom() ve IndicatorCreate() fonksiyonları aracılığıyla belirlermiş dosya ismi ile gösterge kısa ismini birbirine karıştırmayın. Gösterge kısa ismi açık yolla belirtilmemişse, derleme sırasında kaynak kodu içeren dosyanın ismi, kısa isim olarak belirlenecektir.

Bir göstergenin çizelge üzerinden kaldırılması, hesaplama kısmının da terminalde silineceği anlamına gelmez. Göstergenin tanıtıcı değerini serbest bırakmak için IndicatorRelease() fonksiyonunu kullanın.

Gösterge kısa ismi, düzgün şekilde girilmelidir. Bu, INDICATOR_SHORTNAME özelliğine, IndicatorSetString() fonksiyonu kullanılarak yazılır. Kısa isimde, gösterge parametre değerlerinin içerilmesi önerilir; çünkü çizelge penceresinden göstergeyi silerken kullanılan ChartIndicatorDelete() fonksiyonu kısa isimle tanımlanır.

Başlatma başarısız olduktan sonra, göstergenin silinmesi örneği:

```
//+------------------------------------------------------------------+
//|                                  Demo_ChartIndicatorDelete.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             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
//--- giriş parametreleri
input int      first_param=1;
input int      second_param=2;
input int      third_param=3;
input bool     wrong_init=true;
//--- gösterge tamponları
double         HistogramBuffer[];
string         shortname;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int res=INIT_SUCCEEDED;
//--- HistogramBuffer dizisini gösterge dizisine bağla
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- Giriş parametreleri temelinde bir kısa isim belirle
   shortname=StringFormat("Demo_ChartIndicatorDelete(%d,%d,%d)",
                          first_param,second_param,third_param);
   IndicatorSetString(INDICATOR_SHORTNAME,shortname);
//--- Eğer gösterge zorla tamamlanmaya ayarlanmışsa, sıfır olmayan bir değere dönüş ya
   if(wrong_init) res=INIT_FAILED;
   return(res);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
```

```
//--- Döngü içinde çalışmak için başlangıç konumu
   int start=prev_calculated-1;
   if(start<0) start=0;
//--- Gösterge tamponunu değerlerle doldur
   for(int i=start;i<rates_total;i++)
     {
      HistogramBuffer[i]=close[i];
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| A handler of the Deinit event                                    |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   PrintFormat("%s: Sonlandırma sebebi kodu=%d",__FUNCTION__,reason);
   if(reason==REASON_INITFAILED)
     {
      PrintFormat("%s kısa isimli bir gösterge (dosya %s), kendisini çizelgeden sildi"
      int window=ChartWindowFind();
      bool res=ChartIndicatorDelete(0,window,shortname);
      //--- ChartIndicatorDelete() çağrısının sonucunu analiz et
      if(!res)
        {
         PrintFormat("%s göstegesinin #%d penceresinden silinmesi başarısız. Hata kodu
                     shortname,window,GetLastError());
        }
     }
  }
```

Ayrıca Bakınız

ChartIndicatorAdd(), ChartIndicatorName(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

ChartIndicatorGet

Belirtilmiş çizelgede, belirtilmiş isimli göstergenin tanıtıcı değerine dönüş yapar.

```
int  ChartIndicatorGet(
   long           chart_id,              // Çizelge tanımlayıcı
   int            sub_window             // Alt pencerenin numarası
   const string   indicator_shortname    // Gösterge kısa ismi
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

sub_window

[in] Çizelge alt penceresinin numarası. 0, ana çizelge penceresi demektir.

const indicator_shortname

[in] INDICATOR_SHORTNAME özelliği ile IndicatorSetString() fonksiyonu kullanılarak ayarlanan gösterge kısa ismi. Gösterge kısa ismini almak için ChartIndicatorName() fonksiyonunu kullanın.

Dönüş değeri

Başarı durumunda gösterge tanıtıcı değerine, aksi durumda INVALID_HANDLE değerine dönüş yapar. Hata hakkında bilgi almak için, GetLastError() fonksiyonunu çağırın.

Not

ChartIndicatorGet() fonksiyonu kullanılarak elde edilen gösterge tanıtıcı değeri, dahili gösterge kullanım sayacını artırır. Terminal çalışma zamanı sistemi, sayacı sıfırdan büyük olan tüm göstergeleri yüklü tutar. Bu nedenle, artık gerekli olmayan gösterge tanıtıcı değeri, aşağıdaki örnekte gösterildiği gibi, aynı programdaki IndicatorRelease() kullanılarak hemen ve açıkça serbest bırakılmalıdır. Aksi takdirde, "terk edilmiş" tanıtıcı değerini bulmak ve başka bir programdan doğru şekilde serbest bırakmak imkansız olacaktır.

Yeni bir gösterge oluştururken, INDICATOR_SHORTNAME özelliği ile IndicatorSetString() fonksiyonu kullanılarak oluşturulan kısa ismini şekillendirirken dikkatli olun. Göstergenin, ChartIndicatorGet() fonksiyonunda kısa isim ile tanımlanması nedeniyle; kısa ismin gösterge giriş parametrelerinin değerini içermesi önerilir.

Bir göstergeyi tanımlamanın bir diğer yolu - IndicatorParameters() fonksiyonunu kullanarak, verilmiş tanıtıcı değer için gösterge parametrelerinin bir listesini alıp, elde edilen değerleri analiz etmektir.

Örnek:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   //--- Çizelgedeki pencerelerin sayısı (en az bir adet; ana pencere her zaman mevcut
   int windows=(int)ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   //--- Tüm pencereleri kontrol et
```

```
   for(int w=0;w<windows;w++)
     {
      //--- bu penceredeki/alt penceredeki göstergelerin sayısı
      int total=ChartIndicatorsTotal(0,w);
      //--- penceredeki tüm göstergeleri araştır
      for(int i=0;i<total;i++)
        {
         //--- bir göstergenin kısa ismini al
         string name=ChartIndicatorName(0,w,i);
         //--- göstergenin tanıtıcı değerini al
         int handle=ChartIndicatorGet(0,w,name);
         //--- Günlüğe ekle
         PrintFormat("Window=%d,  index=%d,  name=%s,  handle=%d",w,i,name,handle);
         //--- You should obligatorily release the indicator handle when it is no long
         IndicatorRelease(handle);
        }
     }
  }
```

Ayrıca Bakınız

ChartIndicatorAdd(), ChartIndicatorName(), ChartIndicatorsTotal(), IndicatorParameters()

ChartIndicatorName

Belirtilen çizelge penceresinin göstergeler listesindeki numaraları kullanarak, göstergenin kısa ismine dönüş yapar.

```
string  ChartIndicatorName(
   long  chart_id,      // çizelge tanımlayıcısı
   int   sub_window     // alt pencerenin numarası
   int   index          // çizelgenin gösterge listesindeki gösterge indisi
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0 mevcut çizelgeyi ifade eder.

sub_window

[in] Çizelge alt penceresinin indisi. 0 ana çizelge penceresini tanımlar.

index

[in] Göstergenin, gösterge listesindeki indisi. Göstergelerin numaralanması sıfırdan başlar, yani listedeki ilk gösterge 0 indisine sahip olacaktır. Listedeki göstergelerin sayısına ulaşmak için ChartIndicatorsTotal() fonksiyonunu kullanın.

Dönüş değeri

INDICATOR_SHORTNAME özelliği ile IndicatorSetString() fonksiyonu kullanılarak ayarlanan gösterge kısa ismi. Hata detayları için GetLastError() fonksiyonunu kullanın.

Not

Gösterge oluşturulurken iCustom() ve IndicatorCreate() fonksiyonları aracılığıyla belirlermiş dosya ismi ile gösterge kısa ismini birbirine karıştırmayın. Eğer göstergenin ismi açık yoldan ayarlanmamışsa, derleme sırasında, gösterge kaynak kodunu içeren dosyanın ismi, kısa isim olarak belirlenecektir.

Gösterge kısa ismi, düzgün şekilde girilmelidir. Bu, INDICATOR_SHORTNAME özelliğine, IndicatorSetString() fonksiyonu kullanılarak yazılır. Kısa isimde, gösterge parametre değerlerinin içerilmesi önerilir; çünkü çizelge penceresinden göstergeyi silerken kullanılan ChartIndicatorDelete() fonksiyonu kısa isimle tanımlanır.

Ayrıca Bakınız

ChartIndicatorAdd(), ChartIndicatorDelete(), ChartIndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString()

ChartIndicatorsTotal

Belirlenen çizelge penceresine uygulanmış tüm göstergelerin sayısına dönüş yapar.

```
int  ChartIndicatorsTotal(
   long  chart_id,       // çizelge tanımlayıcısı
   int   sub_window      // alt pencerenin numarası
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0 mevcut çizelgeyi ifade eder.

sub_window

[in] Çizelge alt penceresinin indisi. 0 ana çizelge penceresini tanımlar.

Dönüş değeri

Belirlenmiş çizelge penceresindeki gösterge sayısı. Hata detayları için GetLastError() fonksiyonunu kullanın.

Not

Fonksiyon, çizelgeye iliştirilmiş tüm göstergeler üzerinden arama yapılmasını sağlar. Çizelgedeki tüm pencerelerin sayısı, CHART_WINDOWS_TOTAL özelliği ile ChartGetInteger() fonksiyonu kullanılarak elde edilebilir.

Ayrıca Bakınız

ChartIndicatorAdd(), ChartIndicatorDelete(), iCustom(), IndicatorCreate(), IndicatorSetString()

ChartWindowOnDropped

Uzman Danışmanın veya betiğin bırakıldığı çizelge alt penceresinin numarası (indisi). 0, ana çizelge penceresi demektir.

```
int  ChartWindowOnDropped();
```

Dönüş değeri

int tipli değer.

Örnek:

```
   int myWindow=ChartWindowOnDropped();
   int windowsTotal=ChartGetInteger(0,CHART_WINDOWS_TOTAL);
   Print("Script, pencere üzerinde çalışıyor #"+myWindow+
         ". Çizelgedeki toplam pencere sayısı "+ChartSymbol()+":",windowsTotal);
```

Ayrıca Bakınız

ChartPriceOnDropped, ChartTimeOnDropped, ChartXOnDropped, ChartYOnDropped

ChartPriceOnDropped

Uzman Danışmanın veya betiğin bırakıldığı çizelge noktasının koordinatlarına dönüş yapar.

```
double  ChartPriceOnDropped();
```

Dönüş değeri

double tipli değer.

Örnek:

```
   double p=ChartPriceOnDropped();
   Print("ChartPriceOnDropped() = ",p);
```

Ayrıca Bakınız

ChartXOnDropped, ChartYOnDropped

ChartTimeOnDropped

Bir Uzman Danışmanın veya betiğin bırakıldığı çizelge noktasının zaman koordinatına dönüş yapar.

```
datetime  ChartTimeOnDropped();
```

Dönüş değeri

datetime tipi değer.

Örnek:

```
   datetime t=ChartTimeOnDropped();
   Print("Script şuraya bırakıldı "+t);
```

Ayrıca Bakınız

ChartXOnDropped, ChartYOnDropped

ChartXOnDropped

Uzman Danışmanın veya betiğin bırakıldığı çizelge noktasının X koordinatına dönüş yapar.

```
int  ChartXOnDropped();
```

Dönüş değeri

X koordinat değeri.

Not

X ekseninin yönü soldan sağa doğrudur.

Örnek:

```
   int X=ChartXOnDropped();
   int Y=ChartYOnDropped();
   Print("(X,Y) = ("+X+","+Y+")");
```

Ayrıca Bakınız

ChartWindowOnDropped, ChartPriceOnDropped, ChartTimeOnDropped

ChartYOnDropped

Uzman Danışmanın veya betiğin bırakıldığı çizelge noktasının X koordinatına dönüş yapar.

```
int  ChartYOnDropped();
```

Dönüş değeri

Y koordinat değeri.

Not

Y ekseninin yönü yukarıdan aşağıya doğrudur.

Ayrıca Bakınız

ChartWindowOnDropped, ChartPriceOnDropped, ChartTimeOnDropped

ChartSetSymbolPeriod

Belirlenen çizelgenin sembolünü ve periyodunu değiştirir. Fonksiyon asenkrondur, yani komutu gönderir ve çalışmanın tamamlanması için beklemez. Komut çizelge mesajları kuyruğuna eklendi ve önceki komutların işlenmesinin hemen ardından uygulanacak.

```
bool  ChartSetSymbolPeriod(
   long             chart_id,      // Çizelge tanımlayıcısı
   string           symbol,        // Sembol ismi
   ENUM_TIMEFRAMES  period         // Periyot
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

symbol

[in] Çizelge sembolü. NULL değeri mevcut çizelgeyi gösterir (Uzman danışmanın eklendiği)

period

[in] Çizelge periyodu. ENUM_TIMEFRAMES sayımının değerlerinden birini alabilir. 0 mevcut çizelge periyodu anlamına gelir.

Dönüş değeri

Komutun çizelge mesajları kuyruğuna eklenmesi durumunda 'true', aksi durumda 'false' dönüşü yapar. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

Not

Sembol/periyot değişimi çizelgeye eklenmiş Uzman Danışmanın yeniden başlatılmasına yol açar.

Aynı sembol ve periyot değerleri ile yapılan ChartSetSymbolPeriod çağrısı çizelgeyi güncellemek için kullanılabilir (terminaldeki Yenile komutu gibi). Bunun sonucunda çizelgeye eklenmiş olan tüm göstergeler de yeniden hesaplanacaktır. Böylece yeni tik fiyatı gelmeden de göstergeleri hesaplayabilirsiniz (örneğin haftasonları).

Ayrıca Bakınız

ChartSymbol, ChartPeriod

ChartScreenShot

Fonksiyon, çizelgenin mevcut durumunun ekran görüntüsünü GIF, PNG veya BMP formatında alır.

```
bool  ChartScreenShot(
   long             chart_id,                  // Çizelge tanımlayıcı
   string           filename,                  // Sembol ismi
   int              width,                     // Genişlik
   int              height,                    // Yükseklik
   ENUM_ALIGN_MODE  align_mode=ALIGN_RIGHT     // Hizalama tipi
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

filename

[in] Ekran görüntüsü dosyası ismi. 63 Karakteri geçemez. Ekran görüntüsü dosyaları \Files konumunda yer alır.

genişlik

[in] Piksel olarak ekran görüntüsü genişliği.

height

[in] Piksel olarak ekran görüntüsü yüksekliği.

align_mode=ALIGN_RIGHT

[in] Dar ekran görüntüsü çıktı modu. ENUM_ALIGN_MODE sayımının değerlerinden biri. ALIGN_RIGHT, sağa hizala anlamına gelir (sondan çıktı). ALIGN_LEFT sola yaslama demektir.

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Not

Eğer çizelgeden belirli bir konumundan bir ekran görüntüsü almanız gerekiyorsa, öncelikle ChartNavigate() fonksiyonunu kullanarak grafiği konumlandırın. Eğer ekran görüntüsünün yatay büyüklüğü çizelge penceresinden büyükse, align_mode ayarlarına bağlı olarak, çizelgenin ya sağı yada solu çıktılanır.

Örnek:

```
#property description "Bu Uzman Danışman, ChartScreenShot() fonksiyonu kullanılarak, r
#property description "serilerinin nasıl oluşturulabileceğini göstermektedir. Uygunlu
#property description "Görüntülerin genişlik ve yükseklikleri makrolarla belirlenir"

#define        WIDTH  800     // ChartScreenShot() çağrısı için görüntü genişliği
#define        HEIGHT 600     // ChartScreenShot() çağrısı için görüntü yüksekliği

//--- giriş parametreleri
input int      pictures=5;    // Serilerdeki resim sayısı
```

```
int            mode=-1;       // -1 çizelgenin sağ sınırına , 1 - ise sola kaydırmayı
int            bars_shift=300;// ChartNavigate()kullanarak çizelgenin kaydırılması iç
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- Otomatik kaydırılmayı devre dışı bırak
   ChartSetInteger(0,CHART_AUTOSCROLL,false);
//--- Çizelgenin sağ kıyısının kaydırılmasını ayarla
   ChartSetInteger(0,CHART_SHIFT,true);
//--- Mum çizelgesini göster
   ChartSetInteger(0,CHART_MODE,CHART_CANDLES);
//---
   Print("Uzman Danışmanın hazırlanması tamamlandı");
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- Fonksiyonun ismini, çağrı zamanını ve olay işleyiciyi göster
   Print(__FUNCTION__,TimeCurrent(),"   id=",id,"   mode=",mode);
//--- CHARTEVENT_CLICK olayının işleyicisi ("Çizelge üstünde fare tıklaması")
   if(id==CHARTEVENT_CLICK)
     {
      //--- Çizelge kıyısından başlangıç kaydırması
      int pos=0;
      //--- Sol çizelge kenarının işlemi
      if(mode>0)
        {
         //--- Çizelgeyi sol yana kaydır
         ChartNavigate(0,CHART_BEGIN,pos);
         for(int i=0;i<pictures;i++)
           {
            //--- Dosya ismi olarak ve çizelgede göstermek için bir metin ayarla
            string name="ChartScreenShot"+"CHART_BEGIN"+string(pos)+".gif";
            //--- İsmi, çizelge üzerinde yorum olarak göster
```

```
            Comment(name);
            //--- Çizelge görüntüsünü terminal_dizini\MQL5\Files\ konumuna kaydet
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_LEFT))
               Print("Ekran görüntüsünü kaydettik ",name);
            //---
            pos+=bars_shift;
            //--- Kullanıcıya çizelgenin yeni kısmına bakması için zaman tanı
            Sleep(3000);
            //--- Çizelgeyi mevcut konumdan bars_shift çubuk sağa kaydır
            ChartNavigate(0,CHART_CURRENT_POS,bars_shift);
           }
         //--- Modu tersine çevir
         mode*=-1;
        }
      else // Çizelgenin sağ kıyısının işlemi
        {
         //--- Çizelgeyi sağ tarafa kaydır
         ChartNavigate(0,CHART_END,pos);
         for(int i=0;i<pictures;i++)
           {
            //--- Dosya ismi olarak ve çizelgede göstermek için bir metin ayarla
            string name="ChartScreenShot"+"CHART_END"+string(pos)+".gif";
            //--- İsmi, çizelge üzerinde yorum olarak göster
            Comment(name);
            //--- Çizelge görüntüsünü terminal_dizini\MQL5\Files\ konumuna kaydet
            if(ChartScreenShot(0,name,WIDTH,HEIGHT,ALIGN_RIGHT))
               Print("Ekran görüntüsünü kaydettik ",name);
            //---
            pos+=bars_shift;
            //--- Kullanıcıya çizelgenin yeni kısmına bakması için zaman tanı
            Sleep(3000);
            //--- Çizelgeyi mevcut konumdan bars_shift çubuk sağa kaydır
            ChartNavigate(0,CHART_CURRENT_POS,-bars_shift);
           }
         //--- Modu tersine çevir
         mode*=-1;
        }
     }  // CHARTEVENT_CLICK olay işlemesinin sonu
//--- OnChartEvent() olay işleyicisinin sonu
  }
```

Ayrıca Bakınız

ChartNavigate(), Kaynaklar

Alım-Satım Fonksiyonları

Alım-satım faaliyetlerini düzenlemek için tasarlanmış fonksiyonlar grubu.

Alım-satım fonksiyonlarıyla çalışmaya başlamadan önce; emir, işlem ve pozisyon gibi temel kavramların ne olduğunu anlamanız gerekir.

- **Emir**, finansal bir sembolün alınması veya satılması amacıyla aracı kuruma gönderilen komuttur. İki temel emir türü vardır: Piyasa ve Bekleyen. Ek olarak, Kar Al ve Zarar Durdur gibi bazı özel seviyeler de mevcuttur.
- **İşlem**, finansal kıymetin ticari anlamda (alım veya satım) el değiştirmesidir. Alış talep fiyatından (Ask), satış ise arz fiyatından (Bid) gerçekleştirilir. İşlemler piyasa emirlerinin işlenmesi veya bekleyen emirlerin tetiklenmesi sonucu açılabilir. Bazı durumlarda emrin birkaç işlemle sonuçlanabileceğini not edin.
- **Pozisyon** bir alım-satım sorumluluğudur sorumluluğudur (belli bir finansal enstrümandan alınmış veya satılmış sözleşmeler). Uzun pozisyonlar, fiyatların yükseleceği beklentisiyle alınmış finansal menkul kıymetlerdir. Kısa pozisyonlar ise fiyatların düşeceği beklentisiyle açılır ve menkul kıymetin temin edilmesi için bir zorunluluk oluşturur.

Alım-satım işlemleri hakkındaki genel bilgiler müşteri terminali yardımında mevcuttur.

Alım-satım fonksiyonları Uzman Danışmanlarda ve scriptlerde kullanılabilirler. Alım-satım fonksiyonları sadece, Uzman Danışmanın veya betiğin özellikleri içinde "Otomatik alım-satıma izin ver" seçeneği işaretlenmişse çağrılabilirler.

Trading can be allowed or prohibited depending on various factors described in the Trade Permission section.

Fonksiyon	Eylem
OrderCalcMargin	Belirtilen emir tipi için, teminat para birimi cinsinden gereken teminat miktarını hesaplar
OrderCalcProfit	Geçirilen parametrelere göre teminat para birimi cinsinden kar hesaplar
OrderCheck	İstenen alım-satım işlemini gerçekleştirmek için yeterli fon olup olmadığını kontrol eder.
OrderSend	Sunucuya alım-satım istekleri gönderir
OrderSendAsync	Asenkron biçimde, sunucu cevabını beklemeden alım-satım istekleri gönderir
PositionsTotal	Açık pozisyonların sayısına dönüş yapar
PositionGetSymbol	Açık pozisyona karşılık gelen sembol ismine dönüş yapar
PositionSelect	Bir açık pozisyonu daha sonraki çalışmalar için seçer
PositionSelectByTicket	Selects a position to work with by the ticket number specified in it
PositionGetDouble	Açık bir pozisyonun istenen (double) özelliğine dönüş yapar
PositionGetInteger	Açık bir pozisyonun istenen (datetime veya int) özelliğine dönüş yapar

Fonksiyon	Eylem
PositionGetString	Açık bir pozisyonun istenen (string) özelliğine dönüş yapar
PositionGetTicket	Returns the ticket of the position with the specified index in the list of open positions
OrdersTotal	Emir sayısına dönüş yapar
OrderGetTicket	Söz konusu emrin fişine dönüş yapar
OrderSelect	Daha sonraki işler için bir emri seçer
OrderGetDouble	Bir emrin istenen (double) özelliğine dönüş yapar
OrderGetInteger	Bir emrin istenen (datetime veya int) özelliğine dönüş yapar
OrderGetString	Bir emrin istenen (string) özelliğine dönüş yapar
HistorySelect	Sunucu zamanının belirli bir periyodu için, emir ve faaliyet geçmişini düzeltir
HistorySelectByPosition	Belirtilen pozisyon tanımlayıcısı ile işlem geçmişini ister.
HistoryOrderSelect	Daha sonraki işler için geçmişteki bir emri seçer
HistoryOrdersTotal	Geçmişteki emirlerin sayısına dönüş yapar
HistoryOrderGetTicket	Geçmişteki karşılık gelen emrin fişine dönüş yapar
HistoryOrderGetDouble	Geçmişteki bir emrin istenen (double) özelliğine dönüş yapar
HistoryOrderGetInteger	Geçmişteki bir emrin istenen (datetime veya int) özelliğine dönüş yapar
HistoryOrderGetString	Geçmişteki bir emrin istenen (string) özelliğine dönüş yapar
HistoryDealSelect	Uygun fonksiyonlarla yapılacak daha sonraki çağrılar için geçmişteki bir işlemi seçer
HistoryDealsTotal	Geçmişteki işlemlerin sayısına dönüş yapar
HistoryDealGetTicket	Geçmişteki bir işlemin fişine dönüş yapar
HistoryDealGetDouble	Geçmişteki bir işlemin istenen (double) özelliğine dönüş yapar
HistoryDealGetInteger	Geçmişteki bir işlemin istenen (datetime veya int) özelliğine dönüş yapar
HistoryDealGetString	Geçmişteki bir işlemin istenen (string) özelliğine dönüş yapar

OrderCalcMargin

Belirtilen emir tipi için, teminat para birimi cinsinden gereken teminat miktarını hesaplar. Hesaplama mevcut piyasa ortamına göre yapılır, bekleyen emirler ve açık pozisyonlar hesaba katılmaz. Bu planlanan alım-satım işlemi için teminatın değerlendirilmesini sağlar. Değere, hesaptaki para birimi cinsinden dönüş yapılır.

```
bool  OrderCalcMargin(
   ENUM_ORDER_TYPE       action,           // emir tipi
   string                symbol,           // sembol ismi
   double                volume,           // hacim
   double                price,            // açılış fiyatı
   double&               margin            // teminat değerini almak için kullanılacak
   );
```

Parametreler

action

[in] Emir tipi, ENUM_ORDER_TYPE sayımının değerlerinden biri olabilir.

symbol

[in] Sembol ismi.

volume

[in] Alım-satım işleminin hacmi.

price

[in] Açılış fiyatı.

margin

[out] Fonksiyonun başarılı olması durumunda, istenen teminat değerinin yazılacağı değişken. Hesapta herhangi bir bekleyen emir veya açık pozisyon bulunmuyorsa, hesaplama yapılır. Marjin değeri bir çok faktöre bağımlıdır ve farklı piyasa koşullarına göre değişim gösterebilir.

Dönüş değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapacaktır. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

Ayrıca Bakınız

OrderSend(), Emir Özellikleri, Alım-Satım İşlem Tipleri

OrderCalcProfit

Mevcut hesap için kar miktarını hesaplar. Hesaplama, mevcut piyasa ortamına göre ve geçirilen parametreler bazında yapılır. Bu fonksiyon, bir alım-satım işlemin sonucu için ön değerlendirme amacıyla kullanılır. Değere, hesaptaki para birimi cinsinden dönüş yapılır.

```
bool  OrderCalcProfit(
   ENUM_ORDER_TYPE       action,           // emir tipi (ORDER_TYPE_BUY veya ORDER_TYI
   string                symbol,           // sembol ismi
   double                volume,           // hacim
   double                price_open,       // açılış fiyatı
   double                price_close,      // kapanış fiyatı
   double&               profit            // kar değerinin alınacağı değişken
   );
```

Parametreler

action

[in] Emir tipi, ENUM_ORDER_TYPE değerlerinden biri olabilir: ORDER_TYPE_BUY veya ORDER_TYPE_SELL.

symbol

[in] Sembol ismi.

volume

[in] Alım-satım işleminin hacmi.

price_open

[in] Açılış fiyatı.

price_close

[in] Kapanış fiyatı.

profit

[out] Fonksiyonun başarılı olması durumunda, hesaplanan kar değerinin yazılacağı değişken. Hesaplanan kar değeri bir çok faktöre bağlıdır ve farklı piyasa koşullarına göre değişim gösterebilir.

Dönüş değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapacaktır. Geçersiz emir tipi belirtilmişse 'false' dönüşü yapar. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

Ayrıca Bakınız

OrderSend(), Emir Özellikleri, Alım-Satım İşlem Tipleri

OrderCheck

OrderCheck() fonksiyonu, istenen alım-satım işlemini gerçekleştirebilmek için, hesapta yeterli para olup olmadığını kontrol eder. Kontrol sonuçları MqlTradeCheckResult yapısındaki ilgili alanlara yerleştirilir.

```
bool  OrderCheck(
   MqlTradeRequest&       request,      // istek yapısı
   MqlTradeCheckResult&   result        // sonuç yapısı
   );
```

Parametreler

request

[in] İstenen alım-satım işlemini tanımlayan MqlTradeRequest tipi yapının işaretçisi.

result

[in,out] Sonuçların yerleştirileceği MqlTradeCheckResult tipi yapının işaretçisi.

Dönüş değeri

Fon miktarı istenen işlem için yetersizse veya parametreler hatalı doldurulmuşsa 'false' dönüşü yapar. Yapıların (yapı işaretçilerinin) başarıyla kontrol edilmesi durumunda 'true' dönüşü yapar'. **Ama bu, istenen alım-satım işleminin kesin olarak gerçekleşeceği anlamına gelmez.** Fonksiyonun çalıştırılmasının sonuçlarının detaylı açıklamaları için, *result* yapısının alanlarını analiz ediniz.

Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

Ayrıca Bakınız

OrderSend(), Alım-Satım İşlemi Tipleri, Alım-Satım İsteği Yapısı, Alım-Satım İstek Sonucu için Kontrol Yapısı, Bir Alım-Satım İsteği Sonucunun Yapısı

OrderSend

OrderSend() fonksiyonu alım-satım işlemleri için sunucuya istekler göndermek amacıyla kullanılır.

```
bool  OrderSend(
   MqlTradeRequest&  request,      // istek yapısı
   MqlTradeResult&   result        // sonuç yapısı
   );
```

Parametreler

request

[in] Müşterinin alım-satım eylemlerini tarif eden MqlTradeRequest tipi yapının işaretçisi.

result

[in,out] Başarıyla tamamlanması durumunda, bir alım-satım işleminin sonucunu tarif eden MqlTradeResult tipi yapının işaretçisi.

Dönüş değeri

Yapıların basitçe kontrol edilmesi (indis kontrolü) başarılı olmuşsa 'true' değerine dönüş yapar. **Ama bu, alım-satım işleminin başarıyla gerçekleşmesi anlamına gelmez**. Fonksiyonun çalışma sonucuyla ilgili daha detaylı bilgi için, *result* yapısının alanlarını inceleyin.

Not

Alım-satım istekleri, sunucu üzerinde çeşitli kontrol aşamalarından geçerler. Öncelikle, *request* parametresinin gerekli alanlarının doğru şekilde doldurulup doldurulmadığı kontrol edilir. Hata bulunmuyorsa, sunucu, emri daha ileri işlemler için kabul eder. Emir, alım-satım sunucusu tarafından kabul edilmişse, OrderSend() fonksiyonu 'true' dönüşü yapar.

Alım-satım sunucusuna gönderilmeden önce isteğin kontrol edilmesi önerilir. Bir isteği kontrol etmek için OrderCheck() fonksiyonunu kullanabilirsiniz. Bu fonksiyon, alım-satım işlemini gerçekleştirmek için yeterli fon olup olmadığını kontrol eder ve alım satım isteği sonucunun kontrolü içinde kullanışlı parametrelere dönüş yapar:

- dönüş kodu, kontrol edilen istekteki hatalarla ilgili bilgi içerir;
- bakiye değeri, alım-satım işleminin uygulanmasından sonra görünür;
- varlık değeri, alım-satım işleminin uygulanmasından sonra görünür;
- kayan nokta değeri, alım-satım işleminin uygulanmasından sonra görünür;
- alım-satım işlemi için istenen teminat;
- alım-satım işleminin uygulanmasından sonra kalacak olan serbest varlık miktarı;
- alım-satım işleminin uygulanmasından sonra ayarlanacak olan teminat seviyesi;
- dönüş kodu yorumu, hata açıklaması.

Bir piyasa emri gönderirken (MqlTradeRequest.action=TRADE_ACTION_DEAL), OrderSend() fonksiyonunun başarılı sonucu siparişin yürütüldüğü anlamına gelmez (uygun işlemler gerçekleştirilir). Bu durumda 'true', siparişin daha ileri bir şekilde icra edilmesi için trading sisteminde başarılı bir şekilde yerleştirildiği anlamına gelir. Ticaret sunucusu, bir OrderSend() çağrısına tepki oluştururken, bu verilerin farkındaysa, geri döndürülen sonuç yapısındaki *deal veya order* alan değerlerini *doldurabilir*. Genel olarak, bir siparişe karşılık gelen işlemlerin gerçekleştirilme olayı, OrderSend() çağrısına bir yanıt gönderdikten sonra olabilir. Bu nedenle, herhangi bir ticaret talebi türü için, OrderSend() yürütme sonucunu alırken öncelikle *retcode*

ticareti sunucusu yanıt kodunu ve elde edilen sonuç yapısında mevcut olan *retcode_external* harici sistem yanıt kodunu (eğer gerekirse) kontrol etmeliyiz.

Kabul edilen her emir, uygulanmasını gerektirecek koşullardan biri sağlanana kadar, alım-satım sunucusunda bekletilir, bu koşullar şöyle gösterilebilir:

- zaman-aşımı,
- aksi-yönlü isteğin görünmesi,
- işlem fiyatının görülmesiyle emir uygulaması,
- emir iptal isteğinin alınması.

Emrin işlenmesi anında, alım-satım sunucusu tarafından terminale Trade olayının oluşmasıyla ilgili bir mesaj gönderilir - bu olay OnTrade() fonksiyonu ile işlenebilir.

OrderSend() fonksiyonu ile sunucuya gönderilen alım-satım isteğinin işlenme sonucu, OnTradeTransaction işleyicisi ile takip edilebilir. Bir alım-satım isteğinin uygulanması sırasında, OnTradeTransaction işleyicisinin birkaç defa çağrılacağı not edilmelidir.

Örneğin, bir piyasa alım emri gönderirken, bu emir önce işlenir, hesap için uygun bir alım emri oluşturulur, sonra uygulanır ve açık emirler listesinden kaldırılır, ardından emir geçmişine eklenir, uygun bir işlem geçmişe eklenir ve yeni bir pozisyon oluşturulur. OnTradeTransaction fonksiyonu, bu olayların her biri için çağrılacaktır.

Örnek:

```
//--- ORDER_MAGIC değeri
input long order_magic=55555;
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- hesabın deneme hesabı olduğundan emin ol
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("Canlı hesapta script işlemine izin verilmiyor!");
      return;
     }
//--- emri ver veya sil
   if(GetOrdersTotalByMagic(order_magic)==0)
     {
      //--- mevcut emir yok - bir emir ver
      uint res=SendRandomPendingOrder(order_magic);
      Print("Alım-satım sunucusunun dönüş kodu",res);
     }
   else // emirler mevcut - emirleri sil
     {
      DeleteAllOrdersByMagic(order_magic);
     }
//---
  }
//+------------------------------------------------------------------+
```

```
//| Belirtilen ORDER_MAGIC numarasına sahip emirlerin sayısını al    |
//+------------------------------------------------------------------+
int GetOrdersTotalByMagic(long const magic_number)
  {
   ulong order_ticket;
   int total=0;
//--- tüm bekleyen emirleri incele
   for(int i=0;i<OrdersTotal();i++)
      if((order_ticket=OrderGetTicket(i))>0)
         if(magic_number==OrderGetInteger(ORDER_MAGIC)) total++;
//---
   return(total);
  }
//+------------------------------------------------------------------+
//| Belirtilen ORDER_MAGIC numarasına sahip emirlerin sayısını al    |
//+------------------------------------------------------------------+
void DeleteAllOrdersByMagic(long const magic_number)
  {
   ulong order_ticket;
//--- tüm bekleyen emirleri incele
   for(int i=OrdersTotal()-1;i>=0;i--)
      if((order_ticket=OrderGetTicket(i))>0)
         //--- uygun ORDER_MAGIC numaralı bir emir
         if(magic_number==OrderGetInteger(ORDER_MAGIC))
           {
            MqlTradeResult result={};
            MqlTradeRequest request={};
            request.order=order_ticket;
            request.action=TRADE_ACTION_REMOVE;
            OrderSend(request,result);
            //--- sunucu cevabını günlüğe yaz
            Print(__FUNCTION__,": ",result.comment," cevap kodu ",result.retcode);
           }
//---
  }
//+------------------------------------------------------------------+
//| Rassal bir yolla bir bekleyen emir ayarla                        |
//+------------------------------------------------------------------+
uint SendRandomPendingOrder(long const magic_number)
  {
//--- bir istek hazırla
   MqlTradeRequest request={};
   request.action=TRADE_ACTION_PENDING;          // bir bekleyen emir ayarla
   request.magic=magic_number;                   // ORDER_MAGIC
   request.symbol=_Symbol;                       // sembol
   request.volume=0.1;                           // 0.1 lotluk hacim
   request.sl=0;                                 // Stop Loss belirtilmemiş
   request.tp=0;                                 // Take Profit belirtilmemiş
//--- emir tipini şekillendir
```

```
   request.type=GetRandomType();                  // emir tipi
//--- bekleyen emir için fiyatı şekillendir
   request.price=GetRandomPrice(request.type);  // açılış fiyatı
//--- alım-satım isteği gönder
   MqlTradeResult result={};
   OrderSend(request,result);
//--- sunucu cevabını günlüğe yaz
   Print(__FUNCTION__,":",result.comment);
   if(result.retcode==10016) Print(result.bid,result.ask,result.price);
//--- alım-satım sunucusunun cevap kodunu döndür
   return result.retcode;
  }
//+------------------------------------------------------------------+
//| Rassal bir yolla bekleyen emrin tipine dönüş yap                 |
//+------------------------------------------------------------------+
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);
     }
//--- hatalı değer
   return(WRONG_VALUE);
  }
//+------------------------------------------------------------------+
//| Rassal bir yolla ile fiyata dönüş yap                            |
//+------------------------------------------------------------------+
double GetRandomPrice(ENUM_ORDER_TYPE type)
  {
   int t=(int)type;
//--- sembol için durdurma seviyeleri
   int distance=(int)SymbolInfoInteger(_Symbol,SYMBOL_TRADE_STOPS_LEVEL);
//--- son tik verisini al
   MqlTick last_tick={};
   SymbolInfoTick(_Symbol,last_tick);
//--- tipe göre fiyatı hesapla
   double price;
   if(t==2 || t==5) // ORDER_TYPE_BUY_LIMIT veya ORDER_TYPE_SELL_STOP
     {
      price=last_tick.bid; // Bid fiyatından
      price=price-(distance+(MathRand()%10)*5)*_Point;
     }
   else             // ORDER_TYPE_SELL_LIMIT veya ORDER_TYPE_BUY_STOP
     {
```

```
      price=last_tick.ask; // Ask fiyatından
      price=price+(distance+(MathRand()%10)*5)*_Point;
     }
//---
   return(price);
  }
```

Ayrıca Bakınız

Alım-Satım İşlemi Tipleri, Alım-Satım İsteği Yapısı, Alım-Satım İstek Sonucu için Kontrol Yapısı, Bir Alım-Satım İsteği Sonucunun Yapısı

OrderSendAsync

OrderSendAsync() fonksiyonu, alım-satım isteklerini, sunucunun isteğe vereceği cevabı beklemeden, asenkron şekilde göndermek için kullanılır. Bu fonksiyon yüksek frekanslı (algoritma kuralları çerçevesinde, sunucudan gelecek cevabı beklemenin kabul edilemez olduğu) alım-satım yöntemleri için geliştirilmiştir.

```
bool  OrderSendAsync(
   MqlTradeRequest&  request,      // İstek yapısı
   MqlTradeResult&   result        // Sonuç yapısı
   );
```

Parametreler

request

[in] Müşterinin alım-satım eylemini tarif eden MqlTradeRequest tipli yapının işaretçisi.

result

[in,out] Başarıyla tamamlanması (true dönüşü) durumunda, bir alım-satım işleminin sonucunu tarif eden MqlTradeResult tipi yapının işaretçisi.

Dönüş değeri

İstek sunucuya başarıyla gönderilmişse 'true' dönüşü yapar. İstek gönderilememişse 'false' dönüşü yapar. İsteğin gönderilmiş olması durumunda, *result* değişkeni içindeki cevap kodu TRADE_RETCODE_PLACED değerini içerir (kod 10008 - "emir alındı"). Başarılı bir uygulama sadece emrin gönderildiği anlamına gelir ve isteğin alım satım sunucusuna ulaştığını veya işlem için kabul edildiğini garanti etmez. Alınan bir istek işlenirken, alım-satım sunucusu, müşteri terminaline, Trade olayının oluşmasına yol açan (pozisyonların, emirlerin ve işlemlerin durumundaki) değişimler hakkında uyarılar gönderir.

OrderSendAsync() fonksiyonu ile alım-satım sunucusuna gönderilen bir isteğin işlenme sonucu, OnTradeTransaction işleyicisi ile takip edilebilir. Bir alım-satım isteğinin uygulanması sırasında, OnTradeTransaction işleyicisinin birkaç defa çağrılacağı not edilmelidir.

Örneğin, bir piyasa alım emri gönderirken, bu emir önce işlenir, hesap için uygun bir alım emri oluşturulur, sonra uygulanır ve açık emirler listesinden kaldırılır, ardından emir geçmişine eklenir, uygun bir işlem geçmişe eklenir ve yeni bir pozisyon oluşturulur. OnTradeTransaction fonksiyonu, bu olayların her biri için çağrılacaktır. Böyle bir veriyi alabilmek için, fonksiyon parametrelerinin incelenmesi gerekir:

- **trans** - bu parametre, hesaba uygulanan alım-satım faaliyetini tanımlayan MqlTradeTransaction yapısını alır;
- **request** - bu parametre, bir alım-satım faaliyeti içinde sonuçlanmış bir isteği tarif eden MqlTradeRequest yapısını alır;
- **result** - bu parametre, alım-satım isteğinin sonucunu tanımlayan MqlTradeResult yapısını alır.

Not

Bu fonksiyon amacı ve parametreleri açısından OrderSend() fonksiyonu ile aynıdır ama onun aksine asenkron yapıdadır; yani, uygulama sonucunu beklemek için programı duraklatmaz. Bu iki fonksiyonun alım-satım işlem oranlarını, bir Uzman Danışman kullanarak karşılaştırabilirsiniz.

Örnek:

```
#property description "Alım-satım istekleri göndermek için bir Uzman Danışman "
                      " alım-satım istekleri göndermek için bir Uzman Danışman.\r\n"
#property description "OnTrade() ve OnTradeTransaction() fonksiyonlarıyla"
                      " alım-satım olaylarının işlenmesi göstergilmiştir.\r\n"
#property description "Uzman Danışman parametreleri, Magic Number değerinin "
                      " ayarlanmasını ve mesajların Uzmanlar günlüğünde "
#property description "görüntülenmesini sağlar. Versayılan olarak tüm detaylar. görünt
//--- giriş parametreleri
input int  MagicNumber=1234567;      // Uzman Danışman kimliği
input bool DescriptionModeFull=true; // Detaylandırılmış çıktı modu
//--- HistorySelect() çağrısında kullanmak için değişkenler
datetime history_start;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- otomatik alım-satıma izin veriliyor mu kontrol et
   if(!TerminalInfoInteger(TERMINAL_TRADE_ALLOWED))
     {
      Alert("Otomatik alım-satım devre dışı, Uzman Danışman silinecek");
      ExpertRemove();
      return(-1);
     }
//--- gerçek hesapta alım-satım yapılamıyor
   if(AccountInfoInteger(ACCOUNT_TRADE_MODE)==ACCOUNT_TRADE_MODE_REAL)
     {
      Alert("Uzman danışman gerçek hesapta alım-sayım yapamaz!");
      ExpertRemove();
      return(-2);
     }
//--- Bu hesapta alım-satım yapılabilir mi öğren (örneğin, yatırımcı şifresi kullanır
   if(!AccountInfoInteger(ACCOUNT_TRADE_ALLOWED))
     {
      Alert("Bu hesapta alım-satım devre dışı bırakılmış");
      ExpertRemove();
      return(-3);
     }
//--- Uzman Danışmanı çalıştırmak için gereken zamanı alım-satım geçmişini almak için
   history_start=TimeCurrent();
//---
   CreateBuySellButtons();
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
```

```
//--- tüm grafiksel nesneleri sil
   ObjectDelete(0,"Buy");
   ObjectDelete(0,"Sell");
//---
  }
//+------------------------------------------------------------------+
//| TradeTransaction function                                        |
//+------------------------------------------------------------------+
void OnTradeTransaction(const MqlTradeTransaction &trans,
                        const MqlTradeRequest &request,
                        const MqlTradeResult &result)
  {
//--- alım-satım olayının işleyicisinin ismine göre bir başlık
   Print("=> ",__FUNCTION__," zaman: ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- faaliyet tipini, sayım değeri olarak al
   ENUM_TRADE_TRANSACTION_TYPE type=trans.type;
//--- eğer faaliyet, isteğin işlenmesinin bir sonucu ise
   if(type==TRADE_TRANSACTION_REQUEST)
     {
      //--- faaliyet ismini görüntüle
      Print(EnumToString(type));
      //--- işlenen isteğin metin şeklindeki açıklamasını görüntüle
      Print("------------RequestDescription\r\n",
            RequestDescription(request,DescriptionModeFull));
      //--- ve istek sonucunun açıklamasını göster
      Print("------------ ResultDescription\r\n",
            TradeResultDescription(result,DescriptionModeFull));
     }
   else // diğer tipteki faaliyetler için, bu faaliyetin tam açıklamasını görüntüle
     {
      Print("------------ TransactionDescription\r\n",
            TransactionDescription(trans,DescriptionModeFull));
     }
//---
  }
//+------------------------------------------------------------------+
//| Trade function                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- hesap durumunu saklamak için statik değişkenler
   static int prev_positions=0,prev_orders=0,prev_deals=0,prev_history_orders=0;
//--- alım-satım geçmişini iste
   bool update=HistorySelect(history_start,TimeCurrent());
   PrintFormat("HistorySelect(%s , %s) = %s",
               TimeToString(history_start),TimeToString(TimeCurrent()),(string)update);
//--- alım-satım olayının işleyicisinin ismine göre bir başlık
   Print("=> ",__FUNCTION__," zaman: ",TimeToString(TimeCurrent(),TIME_SECONDS));
//--- işleyici ismini ve işleme anındaki emir sayısını görüntüle
```

```
   int curr_positions=PositionsTotal();
   int curr_orders=OrdersTotal();
   int curr_deals=HistoryOrdersTotal();
   int curr_history_orders=HistoryDealsTotal();
//--- emirlerin, pozisyonların, işlemlerin ve değişimlerin (parantez içinde) sayısını
   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);
//--- günlüğü daha uygun şekilde görebilmek için dizgi ayırma karakteri ekle
   Print("");
//--- hesap durumunu sakla
   prev_positions=curr_positions;
   prev_orders=curr_orders;
   prev_deals=curr_deals;
   prev_history_orders=curr_history_orders;
//---
  }
//+------------------------------------------------------------------+
//| ChartEvent function                                              |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- CHARTEVENT_CLICK olayının ("Çizelge üzerinde tıklama") işlenmesi
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      Print("=> ",__FUNCTION__,": sparam = ",sparam);
      //--- minimum işlem hacmi
      double volume_min=SymbolInfoDouble(_Symbol,SYMBOL_VOLUME_MIN);
      //--- "Buy" basılı ise satın al
      if(sparam=="Buy")
        {
         PrintFormat("Al %s %G lot",_Symbol,volume_min);
         BuyAsync(volume_min);
         //--- düğmeyi serbest bırak
         ObjectSetInteger(0,"Buy",OBJPROP_STATE,false);
        }
      //--- "Sell" düğmesi basılı ise sat
      if(sparam=="Sell")
        {
         PrintFormat("Sat %s %G lot",_Symbol,volume_min);
         SellAsync(volume_min);
```

```
         //--- düğmeyi serbest bırak
         ObjectSetInteger(0,"Sell",OBJPROP_STATE,false);
        }
      ChartRedraw();
     }
//---
  }
//+------------------------------------------------------------------+
//| Bir faaliyetin metin açıklamasına dönüş yapar                    |
//+------------------------------------------------------------------+
string TransactionDescription(const MqlTradeTransaction &trans,
                              const bool detailed=true)
  {
//--- Fonksiyonun dönüş yapacağı dizgiyi hazırla
   string desc=EnumToString(trans.type)+"\r\n";
//--- ayrıntılı modda, mümkün olan tüm veriler eklenir
   if(detailed)
     {
      desc+="Sembol: "+trans.symbol+"\r\n";
      desc+="İşlem fişi: "+(string)trans.deal+"\r\n";
      desc+="İşlem tipi: "+EnumToString(trans.deal_type)+"\r\n";
      desc+="Emir fişi: "+(string)trans.order+"\r\n";
      desc+="Emir tipi: "+EnumToString(trans.order_type)+"\r\n";
      desc+="Emir durumu: "+EnumToString(trans.order_state)+"\r\n";
      desc+="Emir zaman tipi: "+EnumToString(trans.time_type)+"\r\n";
      desc+="Emir zaman aşımı tipi: "+TimeToString(trans.time_expiration)+"\r\n";
      desc+="Fiyat: "+StringFormat("%G",trans.price)+"\r\n";
      desc+="Fiyat eşiği: "+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+="Hacim: "+StringFormat("%G",trans.volume)+"\r\n";
     }
//--- alınan dizgiye dönüş yap
   return desc;
  }
//+------------------------------------------------------------------+
//| Bir alım-satım isteğinin metin açıklamasına dönüş yapar          |
//+------------------------------------------------------------------+
string RequestDescription(const MqlTradeRequest &request,
                          const bool detailed=true)
  {
//--- Fonksiyonun dönüş yapacağı dizgiyi hazırla
   string desc=EnumToString(request.action)+"\r\n";
//--- mevcut olan tüm verileri ayrıntılı modda ekle
   if(detailed)
     {
      desc+="Sembol: "+request.symbol+"\r\n";
      desc+="Magic Number (sihirli sayı): "+StringFormat("%d",request.magic)+"\r\n";
      desc+="Emir fişi: "+(string)request.order+"\r\n";
```

```
      desc+="Emir tipi: "+EnumToString(request.type)+"\r\n";
      desc+="Emrin karşılanma: "+EnumToString(request.type_filling)+"\r\n";
      desc+="Emir zaman tipi: "+EnumToString(request.type_time)+"\r\n";
      desc+="Emir zaman aşımı tipi: "+TimeToString(request.expiration)+"\r\n";
      desc+="Fiyat: "+StringFormat("%G",request.price)+"\r\n";
      desc+="Sapma noktaları: "+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+="Hacim: "+StringFormat("%G",request.volume)+"\r\n";
      desc+="Yorum: "+request.comment+"\r\n";
     }
//--- alınan dizgiye dönüş yap
   return desc;
  }
//+------------------------------------------------------------------+
//| İstek işleme sonucunun metin açıklamasına dönüş yapar            |
//+------------------------------------------------------------------+
string TradeResultDescription(const MqlTradeResult &result,
                              const bool detailed=true)
  {
//--- fonksiyonun dönüş yapacağı dizgiyi hazırla
   string desc="Retcode "+(string)result.retcode+"\r\n";
//--- mevcut olan tüm verileri ayrıntılı modda ekle
   if(detailed)
     {
      desc+="İstek kimliği: "+StringFormat("%d",result.request_id)+"\r\n";
      desc+="Emir fişi: "+(string)result.order+"\r\n";
      desc+="İşlem fişi: "+(string)result.deal+"\r\n";
      desc+="Hacim: "+StringFormat("%G",result.volume)+"\r\n";
      desc+="Fiyat: "+StringFormat("%G",result.price)+"\r\n";
      desc+="Ask: "+StringFormat("%G",result.ask)+"\r\n";
      desc+="Bid: "+StringFormat("%G",result.bid)+"\r\n";
      desc+="Yorum: "+result.comment+"\r\n";
     }
//--- alınan dizgiye dönüş yap
   return desc;
  }
//+------------------------------------------------------------------+
//| Alım ve Satım için iki düğme oluştur                             |
//+------------------------------------------------------------------+
void CreateBuySellButtons()
  {
//--- "Buy" isimli nesneyi kontrol et
   if(ObjectFind(0,"Buy")>=0)
     {
      //--- bulunan nesne bir düğme değilse sil
      if(ObjectGetInteger(0,"Buy",OBJPROP_TYPE)!=OBJ_BUTTON)
         ObjectDelete(0,"Buy");
```

```
      }
   else
      ObjectCreate(0,"Buy",OBJ_BUTTON,0,0,0); // "Buy" düğmesini oluştur
//--- "Buy" düğmesini şekillendir
   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);
//--- "Sell" isimli nesnenin varlığını kontrol et
   if(ObjectFind(0,"Sell")>=0)
     {
      //--- bulunan nesne bir düğme değilse sil
      if(ObjectGetInteger(0,"Sell",OBJPROP_TYPE)!=OBJ_BUTTON)
         ObjectDelete(0,"Sell");
     }
   else
      ObjectCreate(0,up_arrow,OBJ_ARROW,0,0,0,0,0);          // Bir ok oluştur
//--- "Buy" düğmesini şekillendir
   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);
//--- düğmeleri hemen görebilmek için çizelgeyi zorla güncele
   ChartRedraw();
//---
  }
//+------------------------------------------------------------------+
//| OrderSendAsync() fonksiyonunu kullanarak al                      |
//+------------------------------------------------------------------+
void BuyAsync(double volume)
  {
//--- isteği hazırla
   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     ="OrderSendAsync() kullanarak Alış";
   MqlTradeResult  res={};
   if(!OrderSendAsync(req,res))
```

```
      {
       Print(__FUNCTION__,": hata ",GetLastError(),", retcode = ",res.retcode);
      }
//---
  }
//+------------------------------------------------------------------+
//| OrderSendAsync() fonksiyonunu kullanarak sat                     |
//+------------------------------------------------------------------+
void SellAsync(double volume)
  {
//--- isteği hazırla
   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     ="OrderSendAsync() kullanarak Satış";
   MqlTradeResult  res={};
   if(!OrderSendAsync(req,res))
     {
      Print(__FUNCTION__,": hata ",GetLastError(),", retcode = ",res.retcode);
     }
//---
  }
//+------------------------------------------------------------------+
```

"Uzmanlar" günlüğünde görüntülenen mesajlara örnek:

```
 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)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   Magic Number (sihirli sayı): 1234567
12:52:52   ExpertAdvisor (EURUSD,H1)   Emir fişi: 16361998
 12:52:52   ExpertAdvisor (EURUSD,H1)   Emir tipi: ORDER_TYPE_SELL
 12:52:52   ExpertAdvisor (EURUSD,H1)   Emrin karşılanma: ORDER_FILLING_FOK
 12:52:52   ExpertAdvisor (EURUSD,H1)   Emir zaman tipi: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Emir zaman aşımı tipi: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Fiyat: 1.29313
12:52:52   ExpertAdvisor (EURUSD,H1)   Sapma noktaları: 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)   Hacim: 0.1
 12:52:52   ExpertAdvisor (EURUSD,H1)   Comment: Sell using 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)   Emir fişi: 16361998
12:52:52   ExpertAdvisor (EURUSD,H1)   İşlem fişi: 15048668
12:52:52   ExpertAdvisor (EURUSD,H1)   Hacim: 0.1
12:52:52   ExpertAdvisor (EURUSD,H1)   Fiyat: 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)   Comment:
 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)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   İşlem fişi: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   İşlem tipi: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Emir fişi: 16361998
 12:52:52   ExpertAdvisor (EURUSD,H1)   Emir tipi: ORDER_TYPE_SELL
 12:52:52   ExpertAdvisor (EURUSD,H1)   Emir durumu: ORDER_STATE_STARTED
 12:52:52   ExpertAdvisor (EURUSD,H1)   Emir zaman tipi: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Emir zaman aşımı tipi: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Fiyat: 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)   Hacim: 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)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   İşlem fişi: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   İşlem tipi: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Emir fişi: 16361998
 12:52:52   ExpertAdvisor (EURUSD,H1)   Emir tipi: ORDER_TYPE_SELL
 12:52:52   ExpertAdvisor (EURUSD,H1)   Emir durumu: ORDER_STATE_STARTED
 12:52:52   ExpertAdvisor (EURUSD,H1)   Emir zaman tipi: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Emir zaman aşımı tipi: 1970.01.01 00:00
```

```
12:52:52   ExpertAdvisor (EURUSD,H1)   Fiyat: 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)   Hacim: 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)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   İşlem fişi: 0
 12:52:52   ExpertAdvisor (EURUSD,H1)   İşlem tipi: DEAL_TYPE_BUY
12:52:52   ExpertAdvisor (EURUSD,H1)   Emir fişi: 16361998
 12:52:52   ExpertAdvisor (EURUSD,H1)   Emir tipi: ORDER_TYPE_SELL
 12:52:52   ExpertAdvisor (EURUSD,H1)   Emir durumu: ORDER_STATE_FILLED
 12:52:52   ExpertAdvisor (EURUSD,H1)   Emir zaman tipi: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Emir zaman aşımı tipi: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Fiyat: 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)   Hacim: 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)   Symbol: EURUSD
12:52:52   ExpertAdvisor (EURUSD,H1)   İşlem fişi: 15048668
 12:52:52   ExpertAdvisor (EURUSD,H1)   İşlem tipi: DEAL_TYPE_SELL
12:52:52   ExpertAdvisor (EURUSD,H1)   Emir fişi: 16361998
 12:52:52   ExpertAdvisor (EURUSD,H1)   Emir tipi: ORDER_TYPE_BUY
 12:52:52   ExpertAdvisor (EURUSD,H1)   Emir durumu: ORDER_STATE_STARTED
 12:52:52   ExpertAdvisor (EURUSD,H1)   Emir zaman tipi: ORDER_TIME_GTC
12:52:52   ExpertAdvisor (EURUSD,H1)   Emir zaman aşımı tipi: 1970.01.01 00:00
12:52:52   ExpertAdvisor (EURUSD,H1)   Fiyat: 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)   Hacim: 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

Açık pozisyonların sayısına dönüş yapar.

```
int  PositionsTotal();
```

Dönüş Değeri

int tipli değer.

Not

Netleştirme (netting) sisteminde (ACCOUNT_MARGIN_MODE_RETAIL_NETTING ve ACCOUNT_MARGIN_MODE_EXCHANGE) bir sembol üzerinde sadece bir pozisyon bulunabilir. Bu pozisyon bir veya daha fazla işlemin sonucu açılmış olabilir. Müşteri terminalinde Araçkutusunun "İşlem" sekmesi içinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Çoklu pozisyonlara izin verilemsi durumunda (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) bir sembol üzerinde birden fazla pozisyon açılabilir.

Ayrıca bakınız

PositionGetSymbol(), PositionSelect(), Pozisyon Özellikleri

PositionGetSymbol

Açık pozisyonun sembol isimine dönüş yapar ve PositionGetDouble, PositionGetInteger, PositionGetString fonksiyonlarıyla yapılabilecek soraki çalışmalar için otomatik olarak pozisyonu seçer.

```
string  PositionGetSymbol(
   int  index      // açık pozisyonlar listesindeki numara
   );
```

Parametreler

index

[in] Pozisyonun, açık pozisyonlar listesindeki numarası.

Dönüş Değeri

string tipli değişken. Pozisyonun bulunamaması durumunda böş bir dizgiye dönüş yapılır. Hata kodu almak için GetLastError() foksiyonunu kullanın.

Not

Netleştirme (netting) sisteminde (ACCOUNT_MARGIN_MODE_RETAIL_NETTING ve ACCOUNT_MARGIN_MODE_EXCHANGE) bir sembol üzerinde sadece bir pozisyon bulunabilir. Bu pozisyon bir veya daha fazla işlemin sonucu açılmış olabilir. Müşteri terminalinde Araçkutusunun "İşlem" sekmesi içinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Çoklu pozisyonlara izin verilemsi durumunda (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) bir sembol üzerinde birden fazla pozisyon açılabilir.

Ayrıca bakınız

PositionsTotal(), PositionSelect(), Pozisyon Özellikleri

PositionSelect

Üzerinde çalışılmak istenen bir açık pozisyonu seçer. İşlem başarıyla tamamlanmışsa 'true', Aksi durumda false dönüşü yapar. Hata hakkında detaylı bilgi almak için GetLastError() çağrısı yapın.

```
bool  PositionSelect(
   string  symbol      // Sembol ismi
   );
```

Parametreler

symbol

[in] finansal Ensrümanın ismi.

Dönüş Değeri

bool tipli değişken.

Not

Netleştirme (netting) sisteminde (ACCOUNT_MARGIN_MODE_RETAIL_NETTING ve ACCOUNT_MARGIN_MODE_EXCHANGE) bir sembol üzerinde sadece bir pozisyon bulunabilir. Bu pozisyon bir veya daha fazla işlemin sonucu açılmış olabilir. Müşteri terminalinde Araçkutusunun "İşlem" sekmesi içinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Çoklu pozisyonlara izin verilemsi durumunda (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) bir sembol üzerinde birden fazla pozisyon açılabilir. Bu durumda PositionSelect fonksiyonu en düşük fiş numarasına sahip olan pozisyonu kapatır.

PositionSelect() bir pozisyonla ilgili verileri program ortamına kopyalar. PositionGetDouble(), PositionGetInteger() ve PositionGetString() çağrıları, daha önceden kopyalanmış bu verilere dönüş yapar. Yani, pozisyonun kendisi artık mevcut olmasa da (veya yön, hacim vb. değiştirilmiş olsa da), ilgili veri hala elde edilebilir. Bu yüzden, güncel verinin alınabilmesi için PositionSelect() çağrısının değişikliklerden sonra yapıldığından emin olmanız gerekir.

Ayrıca bakınız

PositionGetSymbol(), PositionsTotal(), Pozisyon Özellikleri

PositionSelectByTicket

Pozisyon üzerinde belirtilen fiş numarasını kullanarak bir açık pozisyonu seçer. Başarılı ise 'true', aksi durumda 'false' dönüşü yapar. Hata detayları için GetLastError() çağrısını kullanın.

```
bool  PositionSelectByTicket(
   ulong   ticket     // pozisyon fişi
   );
```

Parametreler

ticket

[in] Pozisyon fişi.

Dönüş Değeri

bool tipli bir değişken.

Not

PositionSelectByTicket() foksionu pozisyonun verilerini program ortamına kopyalar. PositionGetDouble(), PositionGetInteger() ve PositionGetString() fonksiyonlarının sonraki çağrıları daha önceden kopyalanmış bu verilere dönüş yapar. Yani, pozisyonun kendisi artık mevcut olmasa da (veya yön, hacim vb. değiştirilmiş olsa da), ilgili veri hala elde edilebilir. En güncel verinin alındığından emin olmak için PositionSelectByTicket() çağrısını verilere erişmeden önce yaptığınızdan emin olmalısınız.

Ayrıca bakınız

PositionGetSymbol(), PositionsTotal(), Pozisyon Özellikleri

PositionGetDouble

PositionGetSymbol veya PositionSelect kullanılarak önceden seçilmiş olan açık pozisyonun bir özelliğine dönüş yapar. Pozisyon özelliği double tipinde olmalıdır. Fonksiyonun iki türü mevcuttur.

1. Hızlı bir şekilde özellik değerine dönüş yapar.

```
double  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id      // özellik tanımlayıcı
   );
```

2. Fonksiyonun başarısına bağlı olarak 'true' veya 'false' değerlerinden birine dönüş yapar. Başarı durumunda özellik değeri referansla geçirilen son parametreye girilir.

```
bool  PositionGetDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  property_id,      // özellik tanımlayıcı
   double&                        double_var        // özellik değerini burada kabul ed
   );
```

Parametreler

property_id

[in] Pozisyon özelliğinin tanımlayıcısı. bu değişkenin ENUM_POSITION_PROPERTY_DOUBLE sayımının değerlerinden birini labilir.

double_var

[out] İstenen özellik değerini alacak olan double tipli değişken.

Dönüş Değeri

double tipli değer. Başarısız sonuç duruunda, 0 dönüşü yapar.

Not

Netleştirme (netting) sisteminde (ACCOUNT_MARGIN_MODE_RETAIL_NETTING ve ACCOUNT_MARGIN_MODE_EXCHANGE) bir sembol üzerinde sadece bir pozisyon bulunabilir. Bu pozisyon bir veya daha fazla işlemin sonucu açılmış olabilir. Müşteri terminalinde Araçkutusunun "İşlem" sekmesi içinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Çoklu pozisyonlara izin verilemsi durumunda (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) bir sembol üzerinde birden fazla pozisyon açılabilir.

Güncel verinin alınabilmesi için PositionSelect() çağrısının değişikliklerden sonra yapıldığından emin olmanız gerekir.

Ayrıca bakınız

PositionGetSymbol(), PositionSelect(), Pozisyon Özellikleri

PositionGetInteger

PositionGetSymbol veya PositionSelect kullanılarak önceden seçilmiş olan açık pozisyonun bir özelliğine dönüş yapar. Pozisyon özelliği datetime veya int tiplerinde olmalıdır. Fonksiyonun iki türü mevcuttur.

1. Hızlı bir şekilde özellik değerine dönüş yapar.

```
long  PositionGetInteger(
   ENUM_POSITION_PROPERTY_INTEGER  property_id      // özellik tanımlayıcı
   );
```

2. Fonksiyonun başarısına bağlı olarak 'true' veya 'false' değerlerinden birine dönüş yapar. Başarı durumunda özellik değeri, referansla geçirilen son parametreye girilir.

```
bool  PositionGetInteger(
   ENUM_POSITION_PROPERTY_INTEGER  property_id,     // özellik tanımlayıcı
   long&                           long_var         // özellik değerini burada kabul e
   );
```

Parametreler

property_id

[in] Pozisyon özelliğinin tanımlayıcısı. Bu değer, ENUM_POSITION_PROPERTY_INTEGER sayımının değerlerinden biri olabilir.

long_var

[out] İstenen özellik değerinin girileceği long tipli değişken.

Dönüş Değeri

long tipli değer. Başarısız sonuç duruunda, 0 dönüşü yapar.

Not

Netleştirme (netting) sisteminde (ACCOUNT_MARGIN_MODE_RETAIL_NETTING ve ACCOUNT_MARGIN_MODE_EXCHANGE) bir sembol üzerinde sadece bir pozisyon bulunabilir. Bu pozisyon bir veya daha fazla işlemin sonucu açılmış olabilir. Müşteri terminalinde Araçkutusunun "İşlem" sekmesi içinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Çoklu pozisyonlara izin verilemsi durumunda (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) bir sembol üzerinde birden fazla pozisyon açılabilir.

Güncel verinin alınabilmesi için PositionSelect() çağrısının değişikliklerden sonra yapıldığından emin olmanız gerekir.

Örnek:

```
//+------------------------------------------------------------------+
//| Alım-satım fonksiyonu                                            |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- pozisyonun varlığını kontrol et ve en son değiştirlme zamanını göster
```

```
   if(PositionSelect(_Symbol))
     {
//--- sonraki işlemler için pozisyon tanımlayıcısını al
      ulong position_ID=PositionGetInteger(POSITION_IDENTIFIER);
      Print(_Symbol," pozisyon #",position_ID);
//--- pozisyonun oluşturulma zamanını 01.01.1970 tarihinden itibaren milisaniyeler cin
      long create_time_msc=PositionGetInteger(POSITION_TIME_MSC);
      PrintFormat("Pozisyon #%d  POSITION_TIME_MSC = %i64 milisaniye => %s",position_I
                  create_time_msc,TimeToString(create_time_msc/1000));
//--- pozisyonun son değiştirilme zamanını 01.01.1970 tarihinden itibaren saniyeler ci
      long update_time_sec=PositionGetInteger(POSITION_TIME_UPDATE);
      PrintFormat("Pozisyon #%d  POSITION_TIME_UPDATE = %i64 saniye => %s",
                  position_ID,update_time_sec,TimeToString(update_time_sec));
//--- pozisyonun son değiştirilme zamanını 01.01.1970 tarihinden itibaren milisaniyele
      long update_time_msc=PositionGetInteger(POSITION_TIME_UPDATE_MSC);
      PrintFormat("Pozisyon #%d  POSITION_TIME_UPDATE_MSC = %i64 milisaniye => %s",
                  position_ID,update_time_msc,TimeToString(update_time_msc/1000));
     }
//---
  }
```

Ayrıca bakınız

PositionGetSymbol(), PositionSelect(), Pozisyon Özellikleri

PositionGetString

PositionGetSymbol veya PositionSelect kullanılarak önceden seçilmiş olan açık pozisyonun bir özelliğine dönüş yapar. İstenen pozisyon özelliği string tipinde olmalıdır. Fonksiyonun iki türü mevcuttur.

1. Hızlı bir şekilde özellik değerine dönüş yapar.

```
string  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id      // Özellik tanımlayıcısı
   );
```

2. Fonksiyonun başarısına bağlı olarak 'true' veya 'false' değerlerinden birine dönüş yapar. Başarı durumunda özellik değeri, referansla geçirilen son parametreye girilir.

```
bool  PositionGetString(
   ENUM_POSITION_PROPERTY_STRING  property_id,      // özellik tanımlayıcı
   string&                        string_var       // özellik değerini burada kabul ed
   );
```

Parametreler

property_id

[in] Pozisyon özelliğinin tanımlayıcısı. Bu değer ENUM_POSITION_PROPERTY_STRING sayımının değerlerinden biri olabilir.

string_var

[out] İstenen özellik değerinin girileceği string tipli değişken.

Dönüş Değeri

string tipli değişken. Fonuksiyonun başarısız olması durumunda boş bir dizgiye dönüş yapılır.

Not

Netleştirme (netting) sisteminde (ACCOUNT_MARGIN_MODE_RETAIL_NETTING ve ACCOUNT_MARGIN_MODE_EXCHANGE) bir sembol üzerinde sadece bir pozisyon bulunabilir. Bu pozisyon bir veya daha fazla işlemin sonucu açılmış olabilir. Müşteri terminalinde Araçkutusunun "İşlem" sekmesi içinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Çoklu pozisyonlara izin verilemsi durumunda (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) bir sembol üzerinde birden fazla pozisyon açılabilir.

Güncel verinin alınabilmesi için PositionSelect() çağrısının değişikliklerden sonra yapıldığından emin olmanız gerekir.

Ayrıca bakınız

PositionGetSymbol(), PositionSelect(), Pozisyon Özellikleri

PositionGetTicket

Açık pozisyonla listesindeki indis numarasına göre bir pozisyonun fişine dönüş yapar ve PositionGetDouble, PositionGetInteger, PositionGetString fonksiyonlarıyla yapılabilecek soraki çalışmalar için otomatik olarak pozisyonu seçer.

```
ulong  PositionGetTicket(
   int  index      // Pozisyonun listedeki numarası
   );
```

Parametreler

index

[in] Pozisyonun açık pozisyonlar listesindeki numarası (numaralandırma 0 ile başlar).

Dönüş Değeri

Pozisyonun fişi. Başarısızlık durumunda 0 dönüşü yapar.

Not

Netleştirme (netting) sisteminde (ACCOUNT_MARGIN_MODE_RETAIL_NETTING ve ACCOUNT_MARGIN_MODE_EXCHANGE) bir sembol üzerinde sadece bir pozisyon bulunabilir. Bu pozisyon bir veya daha fazla işlemin sonucu açılmış olabilir. Müşteri terminalinde Araçkutusunun "İşlem" sekmesi içinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Çoklu pozisyonlara izin verilemsi durumunda (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) bir sembol üzerinde birden fazla pozisyon açılabilir.

Güncel verinin alınabilmesi için PositionSelect() çağrısının değişikliklerden sonra yapıldığından emin olmanız gerekir.

Ayrıca bakınız

PositionGetSymbol(), PositionSelect(), Pozisyon Özellikleri

OrdersTotal

Mevcut emirlerin sayısına dönüş.

```
int  OrdersTotal();
```

Dönüş Değeri

int tipli değer.

Not

Müşteri terminalinde Araçkutusunun "İşlem" sekmesinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır. Emirler alım-satım faaliyetleri gerçekleştirmek için gönderilen isteklerdir, pozisyonlar ise bir veya daha fazla alım-satım işleminin sonucudur.

Netleştirme (netting) sisteminde (ACCOUNT_MARGIN_MODE_RETAIL_NETTING ve ACCOUNT_MARGIN_MODE_EXCHANGE) bir sembol üzerinde sadece bir pozisyon bulunabilir. Bu pozisyon bir veya daha fazla işlemin sonucu açılmış olabilir. Müşteri terminalinde Araçkutusunun "İşlem" sekmesi içinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Çoklu pozisyonlara izin verilemsi durumunda (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) bir sembol üzerinde birden fazla pozisyon açılabilir.

Ayrıca bakınız

OrderSelect(), OrderGetTicket(), EMir özellikleri

OrderGetTicket

Kalanı Pasife Yazarşılık gelen emrin fişine dönüş yapar ve fonksiyonlarla yapılabilecek soraki çalışmalar için otomatik olarak emri seçer.

```
ulong  OrderGetTicket(
   int  index      // Emir listeindeki numara
   );
```

Parametreler

index

[in] Emrin, emirler listesindeki numarası.

Dönüş Değeri

ulong tipli değer. Başarısız sonuç duruunda, 0 dönüşü yapar.

Not

Müşteri terminalinde Araçkutusunun "İşlem" sekmesinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır. Emirler alım-satım faaliyetleri gerçekleştirmek için gönderilen isteklerdir, pozisyonlar ise bir veya daha fazla alım-satım işleminin sonucudur.

Netleştirme (netting) sisteminde (ACCOUNT_MARGIN_MODE_RETAIL_NETTING ve ACCOUNT_MARGIN_MODE_EXCHANGE) bir sembol üzerinde sadece bir pozisyon bulunabilir. Bu pozisyon bir veya daha fazla işlemin sonucu açılmış olabilir. Müşteri terminalinde Araçkutusunun "İşlem" sekmesi içinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Çoklu pozisyonlara izin verilemsi durumunda (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) bir sembol üzerinde birden fazla pozisyon açılabilir.

OrderGetTicket() bir emirle ilgili verileri program ortamına kopyalar. OrderGetDouble(), OrderGetInteger() ve OrderGetString() çağrıları, daha önceden kopyalanmış bu verilere dönüş yapar. Yani, emir artık var olmasa da (veya fiyatı, Zarar Durdur/Kar Al seviyeleri veya zaman aşımı süresi değiştirilmiş olsa bile) emirle ilgili verilere yine de erişilebilir. Emirle ilgili güncel verileri alabilmek için OrderGetTicket() çağrısının değişikliklerden sonra yapıldığından emin olmalısınız.

Örnek:

```
void OnStart()
  {
//--- emir özelliklerinin değerlerini almak için değişkenler
   ulong    ticket;
   double   open_price;
   double   initial_volume;
   datetime time_setup;
   string   symbol;
   string   type;
   long     order_magic;
   long     positionID;
//--- mevcut bekleyen emirlerin sayısı
   uint     total=OrdersTotal();
```

```
//--- emirleri döngü içinde incele
   for(uint i=0;i<total;i++)
     {
      //--- listedeki pozisyona göre emrin fişine dönüş yap
      if((ticket=OrderGetTicket(i))>0)
        {
         //--- emir özelliklerine dönüş yap
         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)));
         //--- emirle ilgili bilgileri hazırla ve göster
         printf("#fiş %d %s %G %s, %G ile %s zamanında başlatıldı",
                ticket,                  // emir fişi
                type,                    // tip
                initial_volume,          // başlangıç hacmi
                symbol,                  // sembol
                open_price,              // belirtilen açılış fiyatı
                TimeToString(time_setup)// emrin girildiği zaman
                );
        }
     }
//---
  }
```

Ayrıca bakınız

OrdersTotal(), OrderSelect(), OrderGetInteger()

OrderSelect

Çalışılmak istenen emri seçer. İşlem başarıyla tamamlanmışsa 'true', aksi durumda 'false' dönüşü yapar. Hata hakkında detaylı bilgi almak için GetLastError() çağrısını kullanın.

```
bool  OrderSelect(
   ulong    ticket      // Emir fişi
   );
```

Parametreler

ticket

[in] Emir fişi.

Dönüş Değeri

bool tipli değişken.

Not

Müşteri terminalinde Araçkutusunun "İşlem" sekmesinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Netleştirme (netting) sisteminde (ACCOUNT_MARGIN_MODE_RETAIL_NETTING ve ACCOUNT_MARGIN_MODE_EXCHANGE) bir sembol üzerinde sadece bir pozisyon bulunabilir. Bu pozisyon bir veya daha fazla işlemin sonucu açılmış olabilir. Müşteri terminalinde Araçkutusunun "İşlem" sekmesi içinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Çoklu pozisyonlara izin verilemsi durumunda (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) bir sembol üzerinde birden fazla pozisyon açılabilir.

OrderGetTicket() fonksiyonu bir emirle ilgili verileri program ortamına kopyalar ve OrderGetDouble(), OrderGetInteger() ve OrderGetString() çağrıları, daha önceden kopyalanmış bu verilere dönüş yapar. Yani, emir artık var olmasa da (veya fiyatı, Zarar Durdur/Kar Al seviyeleri veya zaman aşımı süresi değiştirilmiş olsa bile) emirle ilgili verilere yine de erişilebilir. Emirle ilgili güncel verileri alabilmek için OrderSelect() çağrısının değişikliklerden sonra yapıldığından emin olmalısınız.

Ayrıca bakınız

OrderGetInteger(), OrderGetDouble(), OrderGetString(), OrderCalcProfit(), OrderGetTicket(), Emir Özellikleri

OrderGetDouble

OrderGetTicket veya OrderSelect kullanılarak önceden seçilmiş olan emrin bir özelliğine dönüş yapar. Emir özelliği double tipinde olmalıdır. Fonksiyonun iki türü mevcuttur.

1. Hızlı bir şekilde özellik değerine dönüş yapar.

```
double  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id        // özellik tanımlayıcı
   );
```

2. Fonksiyonun başarı durumuna göre, true (doğru) veya false (yanlış) değerine dönüş yapar. Başarı durumunda özellik değeri, referansla geçirilen son parametreye girilir.

```
bool  OrderGetDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  property_id,       // özellik tanımlayıcı
   double&                     double_var         // özellik değerini burada kabul
   );
```

Parametreler

property_id

[in] Emir özelliğinin tanımlayıcısı. Bu, ENUM_ORDER_PROPERTY_DOUBLE sayımının değerlerinden biri olabilir.

double_var

[out] İstenen özellik değerini alacak olan double tipli değişken.

Dönüş Değeri

double tipli değer. Başarısız sonuç duruunda, 0 dönüşü yapar.

Not

Müşteri terminalinde Araçkutusunun "İşlem" sekmesinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Netleştirme (netting) sisteminde (ACCOUNT_MARGIN_MODE_RETAIL_NETTING ve ACCOUNT_MARGIN_MODE_EXCHANGE) bir sembol üzerinde sadece bir pozisyon bulunabilir. Bu pozisyon bir veya daha fazla işlemin sonucu açılmış olabilir. Müşteri terminalinde Araçkutusunun "İşlem" sekmesi içinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Çoklu pozisyonlara izin verilemsi durumunda (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) bir sembol üzerinde birden fazla pozisyon açılabilir.

Emirle ilgili güncel verileri alabilmek için OrderSelect() çağrısının değişikliklerden sonra yapıldığından emin olmalısınız.

Ayrıca bakınız

OrdersTotal(), OrderGetTicket(), Emir Özellikleri

OrderGetInteger

Bu fonksiyon OrderGetTicket veya OrderSelect kullanılarak önceden seçilmiş olan emrin bir özelliğine dönüş yapar. Emir özelliği datetime veya int tiplerinde olmalıdır. Fonksiyonun iki türü mevcuttur.

1. Hızlı bir şekilde özellik değerine dönüş yapar.

```
long  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id        // özellik tanımlayıcı
   );
```

2. Fonksiyonun başarı durumuna göre, true (doğru) veya false (yanlış) değerine dönüş yapar. Başarı durumunda özellik değeri, referansla geçirilen son parametreye girilir.

```
bool  OrderGetInteger(
   ENUM_ORDER_PROPERTY_INTEGER  property_id,       // özellik tanımlayıcı
   long&                        long_var           // özellik değerini burada kabul ed
   );
```

Parametreler

property_id

[in] Emir özelliğinin tanımlayıcısı. Bu, ENUM_ORDER_PROPERTY_INTEGER sayımının değerlerinden biri olabilir.

long_var

[out] İstenen özellik değerinin girileceği long tipli değişken.

Dönüş Değeri

long tipli değer. Başarısız sonuç duruunda, 0 dönüşü yapar.

Not

Müşteri terminalinde Araçkutusunun "İşlem" sekmesinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Netleştirme (netting) sisteminde (ACCOUNT_MARGIN_MODE_RETAIL_NETTING ve ACCOUNT_MARGIN_MODE_EXCHANGE) bir sembol üzerinde sadece bir pozisyon bulunabilir. Bu pozisyon bir veya daha fazla işlemin sonucu açılmış olabilir. Müşteri terminalinde Araçkutusunun "İşlem" sekmesi içinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Çoklu pozisyonlara izin verilemsi durumunda (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) bir sembol üzerinde birden fazla pozisyon açılabilir.

Emirle ilgili güncel verileri alabilmek için OrderSelect() çağrısının değişikliklerden sonra yapıldığından emin olmalısınız.

Ayrıca bakınız

OrdersTotal(), OrderGetTicket(), Emir Özellikleri

OrderGetString

Bu fonksiyon OrderGetTicket veya OrderSelect kullanılarak önceden seçilmiş olan emrin bir özelliğine dönüş yapar. Emir özelliği dizgi tipinde olmalıdır. Fonksiyonun iki türü mevcuttur.

1. Hızlı bir şekilde özellik değerine dönüş yapar.

```
string  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id        // özellik tanımlayıcı
   );
```

2. Fonksiyonun başarı durumuna göre true (doğru) veya false (yanlış) değerine dönüş yapar. Başarı durumunda özellik değeri, referansla geçirilen son parametreye girilir.

```
bool  OrderGetString(
   ENUM_ORDER_PROPERTY_STRING  property_id,       // özellik tanımlayıcı
   string&                     string_var         // özellik değerini burada kabul edi
   );
```

Parametreler

property_id

[in] Emir özelliğinin tanımlayıcısı. Bu değişken ENUM_ORDER_PROPERTY_STRING sayımının değerlerinden birini alabilir.

string_var

[out] İstenen özellik değerinin girileceği dizgi tipli değişken..

Dönüş Değeri

string tipli değer.

Not

Müşteri terminalinde Araçkutusunun "İşlem" sekmesinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Netleştirme (netting) sisteminde (ACCOUNT_MARGIN_MODE_RETAIL_NETTING ve ACCOUNT_MARGIN_MODE_EXCHANGE) bir sembol üzerinde sadece bir pozisyon bulunabilir. Bu pozisyon bir veya daha fazla işlemin sonucu açılmış olabilir. Müşteri terminalinde Araçkutusunun "İşlem" sekmesi içinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Çoklu pozisyonlara izin verilemsi durumunda (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) bir sembol üzerinde birden fazla pozisyon açılabilir.

Emirle ilgili güncel verileri alabilmek için OrderSelect() çağrısının değişikliklerden sonra yapıldığından emin olmalısınız.

Ayrıca bakınız

OrdersTotal(), OrderGetTicket(), Emir Özellikleri

HistorySelect

Sunucu zamanının belirli bir periyodu için, emir ve işlem geçmişini düzeltir.

```
bool  HistorySelect(
   datetime  from_date,      // Başlangıç tarihi
   datetime  to_date         // Bitiş tarihi
   );
```

Parametreler

from_date

[in] İsteğin başlangıç tarihi.

to_date

[in] İsteğin bitiş tarihi.

Dönüş değeri

Başarı durumunda 'true', aksi durumda 'false' dönüşü yapar.

Not

HistorySelect() fonksiyonu, daha sonraki başvurular için MQL5 programının içinde bir emir listesi ve bir alım-satım listesi oluşturur. İşlem listesinin uzunluğuna HistoryDealsTotal() fonksiyonu ile; emir listesinin uzunluğuna ise HistoryOrdersTotal() fonksiyonu ile dönüş yapılabilir. Emir listesindeki seçimler için HistoryOrderGetTicket() fonksiyonu, işlemler listesinden yapılacak seçimler içinse HistoryDealGetTicket() fonksiyonu daha uyumludur.

HistoryOrderSelect() fonksiyonunu kullandıktan sonra, MQL5 programında mevcut bulunan emir geçmişi sıfırlanır ve emrin fiş ile aranması başarıyla tamamlanmışsa, bulunan emirlerle yeniden doldurulur. Aynısı, MQL5 programında mevcut bulunan işlem geçmişi içinde geçerlidir - HistoryDealSelect() ile sıfırlanır. Sonra işlemi fiş ile arama başarılı olursa yeniden doldurulur.

Örnek:

```
void OnStart()
  {
   color BuyColor =clrBlue;
   color SellColor=clrRed;
//--- alım-satım geçmişini iste
   HistorySelect(0,TimeCurrent());
//--- nesneleri oluştur
   string   name;
   uint     total=HistoryDealsTotal();
   ulong    ticket=0;
   double   price;
   double   profit;
   datetime time;
   string   symbol;
   long     type;
   long     entry;
//--- tüm işlemler için
```

```
   for(uint i=0;i<total;i++)
     {
      //--- işlemlerin fişlerini almayı dene
      if((ticket=HistoryDealGetTicket(i))>0)
        {
         //--- işlem özelliklerini al
         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);
         //--- sadece geçerli sembol için
         if(price && time && symbol==Symbol())
           {
            //--- fiyat nesnesini oluştur
            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);
            //--- nesne özelliklerini ayarla
            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
           }
        }
     }
//--- çizelge üzerinde uygula
   ChartRedraw();
  }
```

Ayrıca Bakınız

HistoryOrderSelect(), HistoryDealSelect()

HistorySelectByPosition

Belirli bir pozisyon tanımlayıcısı ile, emir ve işlem geçmişini düzeltir.

```
bool  HistorySelectByPosition(
   long   position_id     // pozisyon tanımlayıcı - POSITION_IDENTIFIER
   );
```

Parametreler

position_id

[in] Her işlenmiş emir ve her işlem için ayarlanan pozisyon tanımlayıcısı.

Dönüş değeri

Başarı durumunda 'true', aksi durumda 'false' dönüşü yapar.

Not

Alım-satım geçmişindeki emirlerle - "AraçKutusu" çubuğunun "Trade" sekmesinde görünen - bekleyen emirleri birbirine karıştırmayın. İptal edilmiş veya bir faaliyete konu olmuş emirler, "AraçKutusu" çubuğunun "Geçmiş" sekmesinde bulunabilir.

HistorySelectByPosition() fonksiyonu, daha sonraki başvurular için MQL5 programının içinde pozisyon tanımlayıcısı ile belirtilen bir emir listesi ve bir işlem listesi oluşturur. İşlem listesinin uzunluğunu öğrenmek için HistoryDealsTotal() fonksiyonunu, emir listesinin uzunluğunu almak içinse HistoryOrdersTotal() fonksiyonunu kullanın. Emir listesinin elemanlarını gözden geçirmek için HistoryOrderGetTicket() fonksiyonunu, işlem listesinin elemanları içinse HistoryDealGetTicket() fonksiyonunu kullanın.

HistoryOrderSelect() fonksiyonunu kullandıktan sonra, MQL5 programında mevcut bulunan emir geçmişi sıfırlanır ve emrin fiş ile aranması başarıyla tamamlanmışsa, bulunan emirlerle yeniden doldurulur. Aynısı, MQL5 programında mevcut bulunan işlem geçmişi içinde geçerlidir - HistoryDealSelect() ile sıfırlanır. Sonra işlemi fiş ile arama başarılı olursa yeniden doldurulur.

Ayrıca Bakınız

HistorySelect(), HistoryOrderGetTicket(), Emir özellikleri

HistoryOrderSelect

Uygun fonksiyonlarla yapılacak daha sonraki çağrılar için geçmişteki bir emri seçer. Fonksiyon başarıyla tamamlanmışsa 'true', aksi durumda 'false' dönüşü yapar. Hata ile ilgili daha fazla bilgi için GetLastError() fonksiyonunu çağırın.

```
bool  HistoryOrderSelect(
   ulong  ticket      // Emir fişi
   );
```

Parametreler

ticket

[in] Emir fişi.

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Not

Alım-satım geçmişindeki emirlerle - "AraçKutusu" çubuğunun "Trade" sekmesinde görünen - bekleyen emirleri birbirine karıştırmayın. İptal edilmiş veya bir faaliyete konu olmuş emirler, "AraçKutusu" çubuğunun "Geçmiş" sekmesinde bulunabilir.

HistoryOrderSelect() fonksiyonu, bir MQL5 program programı içerisinde, çağrı için mevcut emirlerin listesini temizler ve çalışması başarıyla tamamlanmışsa tek bir emri kopyalar. HistorySelect() fonksiyonu ile seçilmiş tüm emirleri incelemeniz gerekiyorsa, HistoryOrderGetTicket() fonksiyonunu kullanmanız daha iyi olacaktır.

Ayrıca Bakınız

HistorySelect(), HistoryOrderGetTicket(), Emir özellikleri

HistoryOrdersTotal

Geçmişteki emirlerin sayısına dönüş yapar. HistoryOrdersTotal() çağrısını yapmadan önce, HistorySelect() veya HistorySelectByPosition() fonksiyonunu kullanarak emir ve işlem geçmişinin alınması gerekir.

```
int  HistoryOrdersTotal();
```

Dönüş değeri

int tipli değer.

Not

Alım-satım geçmişindeki emirlerle - "AraçKutusu" çubuğunun "Trade" sekmesinde görünen - bekleyen emirleri birbirine karıştırmayın. İptal edilmiş veya bir faaliyete konu olmuş emirler, "AraçKutusu" çubuğunun "Geçmiş" sekmesinde bulunabilir.

Ayrıca Bakınız

HistorySelect(), HistoryOrderSelect(), HistoryOrderGetTicket(), Emir özellikleri

HistoryOrderGetTicket

Geçmişteki karşılık gelen emrin fişine dönüş yapar. HistoryOrderGetTicket() çağrısını yapmadan önce, HistorySelect() veya HistorySelectByPosition() fonksiyonunu kullanarak emir ve işlem geçmişinin alınması gerekir.

```
ulong  HistoryOrderGetTicket(
   int  index      // Emir listesindeki numara
   );
```

Parametreler

index

[in] Emir listesindeki emir numarası.

Dönüş değeri

ulong tipli değer. Fonksiyon, başarısız olması durumunda 0 dönüşü yapar.

Not

Alım-satım geçmişindeki emirlerle - "AraçKutusu" çubuğunun "Trade" sekmesinde görünen - bekleyen emirleri birbirine karıştırmayın. İptal edilmiş veya bir faaliyete konu olmuş emirler, "AraçKutusu" çubuğunun "Geçmiş" sekmesinde bulunabilir.

Örnek:

```
void OnStart()
  {
   datetime from=0;
   datetime to=TimeCurrent();
//--- tüm geçmişi iste
   HistorySelect(from,to);
//--- emir özelliklerinin değerlerini döndürmek için değişkenler
   ulong    ticket;
   double   open_price;
   double   initial_volume;
   datetime time_setup;
   datetime time_done;
   string   symbol;
   string   type;
   long     order_magic;
   long     positionID;
//--- mevcut bekleyen emirlerin sayısı
   uint     total=HistoryOrdersTotal();
//--- döngü içinde incele
   for(uint i=0;i<total;i++)
     {
      //--- listedeki konumuna göre emir fişine dönüş yap
      if((ticket=HistoryOrderGetTicket(i))>0)
        {
         //--- emir özelliklerini döndür
```

```
         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));
         //--- emir bilgisini hazırla ve göster
         printf("#fiş %d %s %G %s %G değerleriyle %s anında işlenmiş => %s anında tama
                ticket,                  // emir fişi
                type,                    // tip
                initial_volume,          // hacim
                symbol,                  // sembol
                open_price,              // belirtilen açılış fiyatı
                TimeToString(time_setup),// emir işleme zamanı
                TimeToString(time_done), // emrin gerçekleşmesinin ve silinmesinin zar
                positionID               // işleme konu olan pozisyonun tanımlayıcısı
                );
        }
     }
//---
  }
//+------------------------------------------------------------------+
//| Emir tipinin ismine dizgi şeklinde dönüş yapar                   |
//+------------------------------------------------------------------+
string GetOrderType(long type)
  {
   string str_type="bilinmeyen işlem";
   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);
  }
```

Ayrıca Bakınız

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Emir özellikleri

HistoryOrderGetDouble

Bir emrin istenen özelliğine dönüş yapar. Emir özelliği double tipinde olmalıdır. Fonksiyonun 2 çeşidi bulunmaktadır.

1. Hemen, özellik değerine dönüş yapar.

```
double  HistoryOrderGetDouble(
   ulong                      ticket_number,     // Fiş
   ENUM_ORDER_PROPERTY_DOUBLE  property_id        // Özellik tanımlayıcı
   );
```

2. Fonksiyonun başarı durumuna göre, 'true' veya 'false' değerine dönüş yapar. Başarı durumunda, özellik değeri, son parametreye referans ile geçirilen hedef değişkene yerleştirilir.

```
bool  HistoryOrderGetDouble(
   ulong                      ticket_number,     // Fiş
   ENUM_ORDER_PROPERTY_DOUBLE  property_id,       // Özellik tanımlayıcı
   double&                    double_var         // Özellik değerini burada farz edi
   );
```

Parametreler

ticket_number

[in] Emir fişi.

property_id

[in] Özellik tanımlayıcı. Bu değer ENUM_ORDER_PROPERTY_DOUBLE sayımının değerlerinden biri olabilir.

double_var

[out] İstenen özellik değerini alacak olan double tipli değişken.

Dönüş değeri

double tipli değer.

Not

Alım-satım geçmişindeki emirlerle - "AraçKutusu" çubuğunun "Trade" sekmesinde görünen - bekleyen emirleri birbirine karıştırmayın. İptal edilmiş veya bir faaliyete konu olmuş emirler, "AraçKutusu" çubuğunun "Geçmiş" sekmesinde bulunabilir.

Ayrıca Bakınız

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Emir özellikleri

HistoryOrderGetInteger

Bir emrin istenen özelliğine dönüş yapar. Emir özelliği datetime veya int tipinde olmalıdır. Fonksiyonun 2 çeşidi bulunmaktadır.

1. Hemen, özellik değerine dönüş yapar.

```
long  HistoryOrderGetInteger(
   ulong                          ticket_number,     // Fiş
   ENUM_ORDER_PROPERTY_INTEGER    property_id        // Özellik tanımlayıcı
   );
```

2. Fonksiyonun başarı durumuna göre, 'true' veya 'false' değerine dönüş yapar. Başarı durumunda, özellik değeri, son parametreye referans ile geçirilen hedef değişkene yerleştirilir.

```
bool  HistoryOrderGetInteger(
   ulong                          ticket_number,     // Fiş
   ENUM_ORDER_PROPERTY_INTEGER    property_id,       // Özellik tanımlayıcı
   long&                          long_var           // Özelik değerini burada farz edi
   );
```

Parametreler

ticket_number

[in] Emir fişi.

property_id

[in] Özellik tanımlayıcı. Bu değer ENUM_ORDER_PROPERTY_INTEGER sayımının değerlerinden biri olabilir.

long_var

[out] İstenen özellik değerini alacak olan long tipli değişken.

Dönüş değeri

long tipli değer.

Not

Alım-satım geçmişindeki emirlerle - "AraçKutusu" çubuğunun "Trade" sekmesinde görünen - bekleyen emirleri birbirine karıştırmayın. İptal edilmiş veya bir faaliyete konu olmuş emirler, "AraçKutusu" çubuğunun "Geçmiş" sekmesinde bulunabilir.

Örnek:

```
//+------------------------------------------------------------------+
//| Trade function                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- Haftanın alım-satım geçmişinden son emrin fişini al
   ulong last_order=GetLastOrderTicket();
   if(HistoryOrderSelect(last_order))
```

```
      {
       //--- emrin işlenmesinin, 01.01.1970 tarihinden buyana geçen milisaniyeler bazır
       long time_setup_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_SETUP_MSC);
       PrintFormat("Emir #%d ORDER_TIME_SETUP_MSC=%i64 => %s",
                   last_order,time_setup_msc,TimeToString(time_setup_msc/1000));
       //--- 01.01.1970 beri geçen milisaniyeler cinsinden emrin gerçekleşme/iptal edi
       long  time_done_msc=HistoryOrderGetInteger(last_order,ORDER_TIME_DONE_MSC);
       PrintFormat("Emir #%d ORDER_TIME_DONE_MSC=%i64 => %s",
                   last_order,time_done_msc,TimeToString(time_done_msc/1000));
      }
    else // başarısızlık durumunda uyar
       PrintFormat("HistoryDealSelect() #%d için başarısız oldu. Hata kodu=%d",
                   last_order,GetLastError());

//---
  }
//+------------------------------------------------------------------+
//| Geçmişteki son emir fişine veya -1 eğerine dönüş yapar           |
//+------------------------------------------------------------------+
ulong GetLastOrderTicket()
  {
//--- son 7 günün geçmişini iste
   if(!GetTradeHistory(7))
     {
      //--- başarısız çağrıda uyar ve -1 dönüşü yap
      Print(__FUNCTION__," HistorySelect() 'false' dönüşü yaptı");
      return -1;
     }
//---
   ulong first_order,last_order,orders=HistoryOrdersTotal();
//--- eğer varsa emirlerle çalış
   if(orders>0)
     {
      Print("Emirler = ",orders);
      first_order=HistoryOrderGetTicket(0);
      PrintFormat("first_order = %d",first_order);
      if(orders>1)
        {
         last_order=HistoryOrderGetTicket((int)orders-1);
         PrintFormat("last_order = %d",last_order);
         return last_order;
        }
      return first_order;
     }
//--- emir bulunamadı, -1 dönüşü yap
   return -1;
  }
//+------------------------------------------------------------------------------+
//| Son yedi günün geçmişini ister ve başarısızlık durumunda 'false' dönüşü yapar |
```

```
//+------------------------------------------------------------------------------+
bool GetTradeHistory(int days)
  {
//--- alım-satım geçmişini istemek için bir haftalık periyot ayarla
   datetime to=TimeCurrent();
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
   ResetLastError();
//--- istek yap ve sonucu kontrol et
   if(!HistorySelect(from,to))
     {
      Print(__FUNCTION__," HistorySelect=false. Hata kodu=",GetLastError());
      return false;
     }
//--- geçmiş başarıyla alındı
   return true;
  }
```

Ayrıca Bakınız

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Emir özellikleri

HistoryOrderGetString

Bir emrin istenen özelliğine dönüş yapar. Emir özelliği string tipinde olmalıdır. Fonksiyonun 2 çeşidi bulunmaktadır.

1. Hemen, özellik değerine dönüş yapar.

```
string  HistoryOrderGetString(
   ulong                      ticket_number,     // Fiş
   ENUM_ORDER_PROPERTY_STRING  property_id       // Özellik tanımlayıcı
   );
```

2. Fonksiyonun başarı durumuna göre, 'true' veya 'false' değerine dönüş yapar. Başarı durumunda, özellik değeri, son parametreye referans ile geçirilen hedef değişkene yerleştirilir.

```
bool  HistoryOrderGetString(
   ulong                      ticket_number,     // Fiş
   ENUM_ORDER_PROPERTY_STRING  property_id,       // Özellik tanımlayıcı
   string&                    string_var         // Özelik değerini burada farz ediyo
   );
```

Parametreler

ticket_number

[in] Emir fişi.

property_id

[in] Özellik tanımlayıcı. Bu değer ENUM_ORDER_PROPERTY_STRING sayımının değerlerinden biri olabilir.

string_var

[out] string tipli değer.

Dönüş değeri

string tipli değer.

Not

Alım-satım geçmişindeki emirlerle - "AraçKutusu" çubuğunun "Trade" sekmesinde görünen - bekleyen emirleri birbirine karıştırmayın. İptal edilmiş veya bir faaliyete konu olmuş emirler, "AraçKutusu" çubuğunun "Geçmiş" sekmesinde bulunabilir.

Ayrıca Bakınız

HistorySelect(), HistoryOrdersTotal(), HistoryOrderSelect(), Emir özellikleri

HistoryDealSelect

Uygun fonksiyonlarla yapılacak daha sonraki çağrılar için geçmişteki bir işlemi seçer. Fonksiyon başarıyla tamamlanmışsa 'true', aksi durumda 'false' dönüşü yapar. Hata ile ilgili daha fazla bilgi için GetLastError() fonksiyonunu çağırın.

```
bool  HistoryDealSelect(
   ulong  ticket      // İşlem fişi
   );
```

Parametreler

ticket

[in] İşlem fişi.

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Not

Emirleri, işlemleri ve pozisyonları birbiriyle karıştırmayın. İşlemler, bir emrin işlenmesinin sonucudur. Her pozisyon, bir veya daha fazla işlemin sonuç özetidir.

HistoryDealSelect() fonksiyonu, bir MQL5 programı içerisinde, referans için mevcut işlemlerin listesini temizler ve başarı durumunda tek bir işlemi kopyalar. HistorySelect() fonksiyonu ile seçilmiş tüm işlemleri incelemeniz gerekiyorsa, HistoryDealGetTicket() fonksiyonunu kullanmanız daha iyi olacaktır.

Ayrıca Bakınız

HistorySelect(), HistoryDealGetTicket(), İşlem Özellikleri

HistoryDealsTotal

Geçmişteki emirlerin sayısına dönüş yapar. HistoryDealsTotal() çağrısını yapmadan önce, HistorySelect() veya HistorySelectByPosition() fonksiyonunu kullanarak emir ve işlem geçmişinin alınması gerekir.

```
int  HistoryDealsTotal();
```

Dönüş değeri

int tipli değer.

Not

Emirleri, işlemleri ve pozisyonları birbiriyle karıştırmayın. İşlemler, bir emrin işlenmesinin sonucudur. Her pozisyon, bir veya daha fazla işlemin sonuç özetidir.

Ayrıca Bakınız

HistorySelect(), HistoryDealGetTicket(), İşlem Özellikleri

HistoryDealGetTicket

Sonraki eylemler için bir işlem seçer ve işlemin geçmiş üzerindeki fişine dönüş yapar. HistoryDealGetTicket() çağrısını yapmadan önce, HistorySelect() veya HistorySelectByPosition() fonksiyonunu kullanarak emir ve işlem geçmişinin alınması gerekir.

```
ulong  HistoryDealGetTicket(
   int  index      // işlem fişi
   );
```

Parametreler

index

[in] İşlemin işlemler listesindeki numarası

Dönüş değeri

ulong tipli değer. Fonksiyon, başarısız olması durumunda 0 dönüşü yapar.

Not

Emirleri, işlemleri ve pozisyonları birbiriyle karıştırmayın. İşlemler, bir emrin işlenmesinin sonucudur. Her pozisyon, bir veya daha fazla işlemin sonuç özetidir.

Örnek:

```
void OnStart()
  {
   ulong deal_ticket;            // işlem fişi
   ulong order_ticket;           // işlemin gerçekleştiği emrin fişi
   datetime transaction_time;    // işlemin gerçekleşme zamanı
   long deal_type ;              // alım-satım işleminin tipi
   long position_ID;             // pozisyon tanımlayıcısı
   string deal_description;      // işlem tanımlayıcısı
   double volume;                // işlem hacmi
   string symbol;                // işlemin yapıldığı sembol
//--- işlem geçmişinin isteneceği aralığın başlangıç ve bitiş tarihlerini ayarla
   datetime from_date=0;         // en baştan
   datetime to_date=TimeCurrent();// şu ana kadar
//--- belirtilen aralıktaki işlem geçmişini iste
   HistorySelect(from_date,to_date);
//--- işlem listesindeki toplam sayı
   int deals=HistoryDealsTotal();
//--- şimdi her bir alım-satımı işle
   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
      //--- işlem numarası için biçimlendirmeyi ayarla
      string print_index=StringFormat("% 3d",i);
      //--- işlem bilgisini göster
      Print(print_index+": deal #",deal_ticket," işlem zamanı ve açıklaması: ",transa
     }
  }
//+------------------------------------------------------------------+
//| Işlemin açıklamasına dizgi şeklinde dönüş yapar                  |
//+------------------------------------------------------------------+
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");
      case DEAL_TYPE_COMMISSION_AGENT_MONTHLY: return ("monthly agent commission");
      case DEAL_TYPE_INTEREST:                 return ("interest rate");
      case DEAL_TYPE_BUY_CANCELED:             descr="cancelled buy deal"; break;
      case DEAL_TYPE_SELL_CANCELED:            descr="cancelled sell deal"; break;
     }
   descr=StringFormat("%s %G %s (emir #%d, pozisyon tanıtıcı %d)",
                      descr,  // mevcut açıklama
                      volume, // işlem hacmi
                      symbol, // işlem sembolü
                      ticket, // işlem emrinin fişi
                      pos_ID  // işlemin dahil olduğu pozisyonun tanımlayıcısı
                      );
   return(descr);
//---
  }
```

Ayrıca Bakınız

HistorySelect(), HistoryDealsTotal(), HistoryDealSelect(), İşlem Özellikleri

HistoryDealGetDouble

Bir işlemin istenen özelliğine dönüş yapar. İşlem özelliği double tipinde olmalıdır. Fonksiyonun 2 çeşidi bulunmaktadır.

1. Hemen, özellik değerine dönüş yapar.

```
double  HistoryDealGetDouble(
   ulong                      ticket_number,     // Fiş
   ENUM_DEAL_PROPERTY_DOUBLE  property_id        // Özellik tanımlayıcı
   );
```

2. Fonksiyonun başarı durumuna göre, 'true' veya 'false' değerine dönüş yapar. Başarı durumunda, özellik değeri, son parametreye referans ile geçirilen hedef değişkene yerleştirilir.

```
bool  HistoryDealGetDouble(
   ulong                      ticket_number,     // Fiş
   ENUM_DEAL_PROPERTY_DOUBLE  property_id,       // Özellik tanımlayıcı
   double&                    double_var         // Özellik değerini burada farz ediyo
   );
```

Parametreler

ticket_number

[in] İşlem fişi.

property_id

[in] İşlem özelliğinin tanımlayıcısı. Bu değer ENUM_DEAL_PROPERTY_DOUBLE sayımının değerlerinden biri olabilir.

double_var

[out] İstenen özellik değerini alacak olan double tipli değişken.

Dönüş değeri

double tipli değer.

Not

Emirleri, işlemleri ve pozisyonları birbiriyle karıştırmayın. İşlemler, bir emrin işlenmesinin sonucudur. Her pozisyon, bir veya daha fazla işlemin sonuç özetidir.

Ayrıca Bakınız

HistorySelect(), HistoryDealsTotal(), HistoryDealSelect(), İşlem Özellikleri

HistoryDealGetInteger

Bir işlemin istenen özelliğine dönüş yapar. İşlem özelliği datetime veya int tipinde olmalıdır. Fonksiyonun 2 çeşidi bulunmaktadır.

1. Hemen, özellik değerine dönüş yapar.

```
long  HistoryDealGetInteger(
   ulong                      ticket_number,     // Fiş
   ENUM_DEAL_PROPERTY_INTEGER  property_id       // Özellik tanımlayıcısı
   );
```

2. Fonksiyonun başarı durumuna göre, 'true' veya 'false' değerine dönüş yapar. Başarı durumunda, özellik değeri, son parametreye referans ile geçirilen hedef değişkene yerleştirilir.

```
bool  HistoryDealGetInteger(
   ulong                      ticket_number,     // Fiş
   ENUM_DEAL_PROPERTY_INTEGER  property_id,       // Özellik tanımlayıcısı
   long&                      long_var           // Özelik değerini burada farz ediyo
   );
```

Parametreler

ticket_number

[in] Alım-satım fişi.

property_id

[in] Özellik tanımlayıcı. Bu değer, ENUM_DEAL_PROPERTY_INTEGER sayımının değerlerinden biri olabilir.

long_var

[out] İstenen özellik değerini alacak olan long tipli değişken.

Dönüş değeri

long tipli değer.

Not

Emirleri, işlemleri ve pozisyonları birbiriyle karıştırmayın. İşlemler, bir emrin işlenmesinin sonucudur. Her pozisyon, bir veya daha fazla işlemin sonuç özetidir.

Örnek:

```
//+------------------------------------------------------------------+
//| Trade function                                                   |
//+------------------------------------------------------------------+
void OnTrade()
  {
//--- haftanın alım-satım geçmişinden, son işlemin fişini al
   ulong last_deal=GetLastDealTicket();
   if(HistoryDealSelect(last_deal))
     {
```

```
      //--- işlemin 01.01.1970 tarihinden buyana geçen milisaniyeler bazındaki zamanı
      long deal_time_msc=HistoryDealGetInteger(last_deal,DEAL_TIME_MSC);
      PrintFormat("İşlem #%d DEAL_TIME_MSC=%i64 => %s",
                  last_deal,deal_time_msc,TimeToString(deal_time_msc/1000));
     }
   else
      PrintFormat("HistoryDealSelect() #%d için başarısız oldu. Hata kodu=%d",
                  last_deal,GetLastError());
//---
  }
//+------------------------------------------------------------------+
//| Geçmişteki son işlemin fişine veya -1 değerine dönüş yapar       |
//+------------------------------------------------------------------+
ulong GetLastDealTicket()
  {
//--- son 7 günün geçmişini iste
   if(!GetTradeHistory(7))
     {
      //--- başarısız çağrıda uyar ve -1 dönüşü yap
      Print(__FUNCTION__," HistorySelect() 'false' dönüşü yaptı");
      return -1;
     }
//---
   ulong first_deal,last_deal,deals=HistoryOrdersTotal();
//--- eğer varsa emirlerle çalış
   if(deals>0)
     {
      Print("İşlemler = ",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;
     }
//--- işlem bulunamadı, -1 dönüşü yap
   return -1;
  }
//+------------------------------------------------------------------------------------+
//| Son yedi günün geçmişini ister ve başarısızlık durumunda 'false' dönüşü yapar |
//+------------------------------------------------------------------------------------+
bool GetTradeHistory(int days)
  {
//--- alım-satım geçmişini istemek için bir haftalık periyot ayarla
   datetime to=TimeCurrent();
   datetime from=to-days*PeriodSeconds(PERIOD_D1);
```

```
   ResetLastError();
//--- istek yap ve sonucu kontrol et
   if(!HistorySelect(from,to))
     {
      Print(__FUNCTION__," HistorySelect=false. Hata kodu=",GetLastError());
      return false;
     }
//--- geçmiş başarıyla alındı
   return true;
  }
```

Ayrıca Bakınız

HistoryDealsTotal(), HistorySelect(), HistoryDealGetTicket(), İşlem Özellikleri

HistoryDealGetString

Bir işlemin istenen özelliğine dönüş yapar. İşlemin özelliği string tipinde olmalıdır. Fonksiyonun 2 çeşidi bulunmaktadır.

1. Hemen, özellik değerine dönüş yapar.

```
string  HistoryDealGetString(
   ulong                      ticket_number,     // Fiş
   ENUM_DEAL_PROPERTY_STRING  property_id        // Özellik tanımlayıcı
   );
```

2. Fonksiyonun başarı durumuna göre, 'true' veya 'false' değerine dönüş yapar. Başarı durumunda, özellik değeri, son parametreye referans ile geçirilen hedef değişkene yerleştirilir.

```
bool  HistoryDealGetString(
   ulong                      ticket_number,     // Fiş
   ENUM_DEAL_PROPERTY_STRING  property_id,       // Özellik tanımlayıcı
   string&                    string_var         // Özelik değerini burada farz ediyo
   );
```

Parametreler

ticket_number

[in] İşlem fişi.

property_id

[in] Özellik tanımlayıcı. Bu değer ENUM_DEAL_PROPERTY_STRING sayımının değerlerinden biri olabilir.

string_var

[out] İstenen özellik değerini alacak olan string tipli değişken.

Dönüş değeri

string tipli değer.

Not

Emirleri, işlemleri ve pozisyonları birbiriyle karıştırmayın. İşlemler, bir emrin işlenmesinin sonucudur. Her pozisyon, bir veya daha fazla işlemin sonuç özetidir.

Ayrıca Bakınız

HistoryDealsTotal(), HistorySelect(), HistoryDealGetTicket(), İşlem Özellikleri

Alım-Satım Sinyalleri

Bu fonksiyon grubu alım-atım sinyallerini yönetebilmek için tasarlanmıştır. Fonksiyonlar ile şunlar yapılabilir:

- kopyalanabilir alım-satım sinyalleri hakkında bilgi edinme,
- sinyal kopyalama ayarlarının alınması ve ayarlanması,
- MQL5 fonksiyonları kullanarak sinyallere abone olma veya aboneliği sonlandırma.

Fonksiyon	Eylem
SignalBaseGetDouble	Seçilen sinyalin double tipli bir özelliğine dönüş yapar
SignalBaseGetInteger	Seçilen sinyalin tamsayı tipli bir özelliğine dönüş yapar
SignalBaseGetString	Seçilen sinyalin string tipli bir özelliğine dönüş yapar
SignalBaseSelect	Terminalde mevcut olan sinyaller arasından çalışılacak sinyali seçer
SignalBaseTotal	Terminaldeki mevcut sinyallerin toplam sayısına dönüş yapar
SignalInfoGetDouble	Sinyal kopyalama ayarlarının double tipli bir özelliğine dönüş yapar
SignalInfoGetInteger	Sinyal kopyalama ayarlarının tamsayı tipli bir özelliğine dönüş yapar
SignalInfoGetString	Sinyal kopyalama ayarlarının string tipli bir özelliğine dönüş yapar
SignalInfoSetDouble	Sinyal kopyalama ayarlarının double tipli bir özelliğinin değerini ayarlar
SignalInfoSetInteger	Sinyal kopyalama ayarlarının double tipli bir özelliğine dönüş yapar
SignalSubscribe	Bir alım-satım sinyaline abone olur
SignalUnsubscribe	Aboneliği sonlandırır

SignalBaseGetDouble

Seçilen sinyalin double tipli özellik değerine dönüş yapar.

```
double  SignalBaseGetDouble(
   ENUM_SIGNAL_BASE_DOUBLE     property_id,     // özellik tanımlayıcı
   );
```

Parametreler

property_id

[in] Sinyal özelliğinin tanımlayıcısı. Bu değişken ENUM_SIGNAL_BASE_DOUBLE sayımının değerlerinden birini alabilir.

Dönüş Değeri

Seçili sinyalin double tipli bir özelliğinin değeri.

SignalBaseGetInteger

Seçilen sinyalin tamsayı tipli özellik değerine dönüş yapar.

```
long  SignalBaseGetInteger(
   ENUM_SIGNAL_BASE_INTEGER     property_id,     // özellik tanımlayıcı
   );
```

Parametreler

property_id

[in] Sinyal özelliğinin tanımlayıcısı. Bu değişken, ENUM_SIGNAL_BASE_INTEGER sayımının değerlerinden birini alabilir.

Dönüş Değeri

Seçili sinyalin tamsayı tipli bir özelliğinin değeri.

SignalBaseGetString

Seçilen sinyalin string tipli özellik değerine dönüş yapar.

```
string  SignalBaseGetString(
   ENUM_SIGNAL_BASE_STRING     property_id,     // özellik tanımlayıcı
   );
```

Parametreler

property_id

[in] Sinyal özelliğinin tanımlayıcısı. Bu değişken, ENUM_SIGNAL_BASE_STRING sayımının değerlerinden birini alabilir.

Dönüş Değeri

Seçilen sinyalin string tipli özellik değerine dönüş yapar.

SignalBaseSelect

Terminalde mevcut olan sinyaller arasından çalışılacak sinyali seçer.

```
bool  SignalBaseSelect(
   int      index      // sinyal numarası
   );
```

Parametreler

index

[in] Sinyalin alım-satım sinyalleri arasındaki numarası.

Dönüş Değeri

Başarı durumunda 'true', aksi durumda 'false' dönüşü yapar. Hata durumuyla ilgili daha fazla bilgi için GetLastError() çağrısını yapın.

Örnek:

```
void OnStart()
  {
//--- termnaldeki toplam sinyal sayısını al
   int total=SignalBaseTotal();
//--- tüm sinyalleri işle
   for(int i=0;i<total;i++)
     {
      //--- sinyali indisine göre seç
      if(SignalBaseSelect(i))
        {
         //--- sinyal özelliklerini al
         long   id    =SignalBaseGetInteger(SIGNAL_BASE_ID);          // sinyal tanım
         long   pips  =SignalBaseGetInteger(SIGNAL_BASE_PIPS);        // pip cinsinde
         long   subscr=SignalBaseGetInteger(SIGNAL_BASE_SUBSCRIBERS); // abone sayısı
         string name  =SignalBaseGetString(SIGNAL_BASE_NAME);         // sinyal ismi
         double price =SignalBaseGetDouble(SIGNAL_BASE_PRICE);        // sinyal fiyat
         string curr  =SignalBaseGetString(SIGNAL_BASE_CURRENCY);     // sinyalin döv
         //--- karlı ve aboenelere sahip olan tüm ücretsiz sinyalleri yazdır
         if(price==0.0 && pips>0 && subscr>0)
            PrintFormat("tanımlayıcı=%d, isim=\"%s\", döviz=%s, pip=%d, aboneler=%d",
        }
      else PrintFormat("SignalBaseSelect hatası. Hata kodu=%d",GetLastError());
     }
  }
```

SignalBaseTotal

Terminaldeki mevcut sinyallerin toplam sayısına dönüş yapar.

```
int  SignalBaseTotal();
```

Dönüş Değeri

Terminaldeki mevcut sinyallerin toplam sayısı.

SignalInfoGetDouble

Sinyal kopyalama ayarlarının double tipli özelliğine dönüş yapar.

```
double  SignalInfoGetDouble(
   ENUM_SIGNAL_INFO_DOUBLE     property_id,     // özellik tanımlayıcı
   );
```

Parametreler

property_id

[in] Sinyal kopyalama ayarları özelliğinin tanımlayıcısı. Bu değişken, ENUM_SIGNAL_INFO_DOUBLE sayımının değerlerinden birini alabilir.

Dönüş Değeri

Sinyal kopyalama ayarlarının double tipli bir özelliği.

SignalInfoGetInteger

Sinyal kopyalama ayarlarının tamsayı tipli özelliğine dönüş yapar.

```
long  SignalInfoGetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id,     // özellik tanımlayıcı
   );
```

Parametreler

property_id

[in] Sinyal kopyalama ayarları özelliğinin tanımlayıcısı. Bu değişken, ENUM_SIGNAL_INFO_INTEGER sayımının değerlerinden birini alabilir.

Dönüş Değeri

Sinyal kopyalama ayarlarının tamsayı tipli bir özelliği.

SignalInfoGetString

Sinyal kopyalama ayarlarının string tipli bir özelliğine dönüş yapar.

```
string  SignalInfoGetString(
   ENUM_SIGNAL_INFO_STRING     property_id,     // özellik tanımlayıcı
   );
```

Parametreler

property_id

[in] Sinyal kopyalama ayarları özelliğinin tanımlayıcısı. Bu değişken ENUM_SIGNAL_INFO_STRING sayımının değerlerinden birini alabilir.

Dönüş Değeri

Sinyal kopyalama ayarlarının string tipli bir özelliği.

SignalInfoSetDouble

Sinyal kopyalama ayarlarının double tipli bir özelliğinin değerini ayarlar.

```
bool  SignalInfoSetDouble(
   ENUM_SIGNAL_INFO_DOUBLE      property_id,     // özellik tanımlayıcı
   double                       value            // yeni değer
   );
```

Parametreler

property_id

[in] Sinyal kopyalama ayarları özelliğinin tanımlayıcısı. Bu değişken, ENUM_SIGNAL_INFO_DOUBLE sayımının değerlerinden birini alabilir.

value

[in] sinyal kopyalama ayarının yeni değeri.

Dönüş Değeri

Özellik değeri değiştirilmişse 'true', aksi durumda 'false' dönüşü yapar. Hata durumuyla ilgili daha fazla bilgi için GetLastError() çağrısını yapın.

SignalInfoSetInteger

Sinyal kopyalama ayarlarının tamsayı tipli bir özelliğinin değerini ayarlar.

```
bool  SignalInfoSetInteger(
   ENUM_SIGNAL_INFO_INTEGER     property_id,     // özellik tanımlayıcı
   long                         value            // yeni değer
   );
```

Parametreler

property_id

[in] Sinyal kopyalama ayarları özelliğinin tanımlayıcısı. Bu değişken, ENUM_SIGNAL_INFO_INTEGER sayımının değerlerinden birini alabilir.

value

[in] sinyal kopyalama ayarının yeni değeri.

Dönüş Değeri

Özellik değeri değiştirilmişse 'true', aksi durumda 'false' dönüşü yapar. Hata durumuyla ilgili daha fazla bilgi için GetLastError() çağrısını yapın.

SignalSubscribe

Bir alım-satım sinyaline abone olur.

```
bool  SignalSubscribe(
   long     signal_id     // sinyal tanımlayıcı
   );
```

Parametreler

signal_id

[in] Sinyal tanımlayıcı.

Dönüş Değeri

Abonelik işlemi başarılı ise 'true', aksi durumda ise 'false' değerine dönüş yapar. Hata durumuyla ilgili daha fazla bilgi için GetLastError() çağrısını yapın.

SignalUnsubscribe

Aboneliği sonlandırır.

```
bool  SignalUnsubscribe();
```

Dönüş Değeri

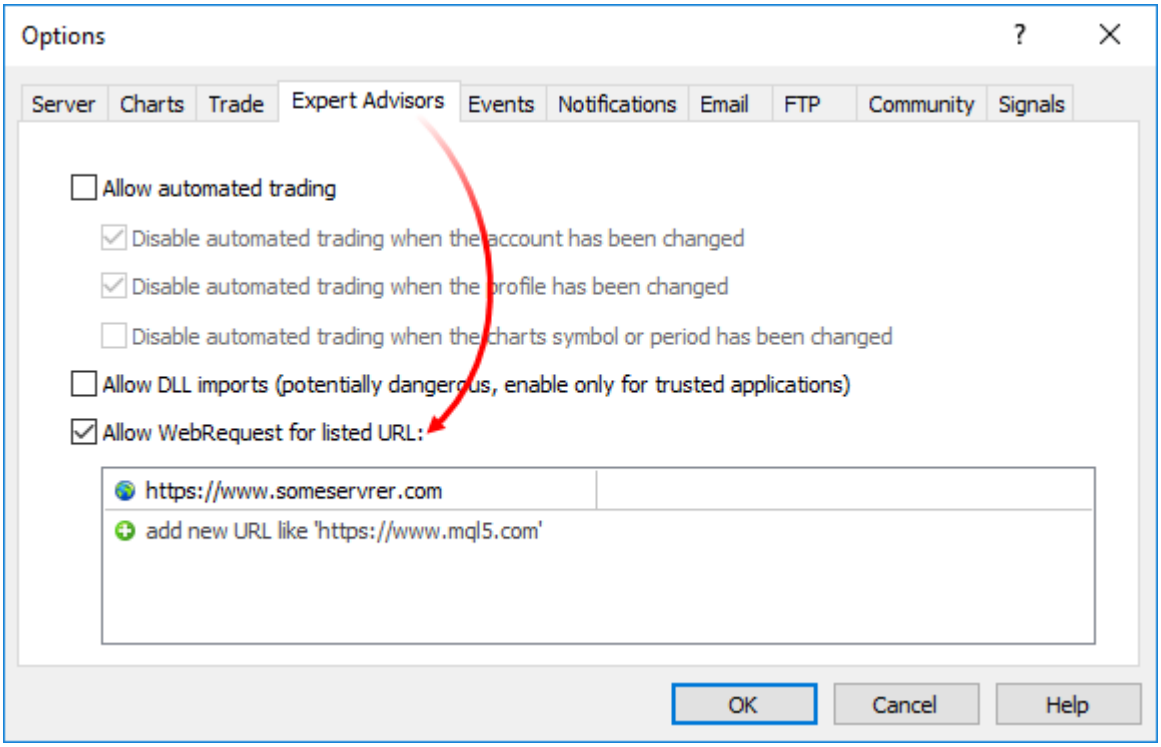
Aboneliği sonlandırma işlemi başarılı ise 'true', aksi durumda ise 'false' değerine dönüş yapar. Hata durumuyla ilgili daha fazla bilgi için GetLastError() çağrısını yapın.

Ağ fonksiyonları

MQL5 programları uzak sunucularla veri alışverişi yapabilmenin yanı sıra, anlık bildirimler, e-postalar ve verileri FTP üzerinden gönderebilir.

- Socket* fonksiyon grubu, uzak bir ana bilgisayarla sistem soketleri üzerinden bir TCP bağlantısı (güvenli bir TLS dahil) kurulmasını sağlar. Çalışma prensibi basittir: bir soket oluşturun, sunucuya bağlanın ve veri okuma ve yazmaya başlayın.
- WebRequest fonksiyonu, web kaynaklarıyla çalışmak üzere tasarlanmıştır ve HTTP isteklerinin (GET ve POST dahil) kolayca gönderilmesini sağlar.
- SendFTP, SendMail ve SendNotification dosyalar, e-postalar ve mobil bildirimler göndermek için daha basit fonksiyonlardır.

Son kullanıcı güvenliği için, izin verilen IP adreslerinin listesi müşteri terminali tarafında gerçekleştirilir. Liste, MQL5 programının Socket* ve WebRequest fonksiyonları aracılığıyla bağlanmasına izin verilen IP adreslerini içerir. Örneğin, programın https://www.someserver.com adresine bağlanması gerekiyorsa, bu adres listede bir terminal kullanıcısı tarafından açıkça belirtilmelidir. Bir adres programsal olarak eklenemez.



Bir kullanıcıya ek yapılandırma ihtiyacı olduğunu bildirmek için MQL5 programına açık bir mesaj ekleyin. Bunu #property description, Alert veya Print aracılığıyla yapabilirsiniz.

Fonksiyon	Eylem
SocketCreate	Belirtilen bayraklara sahip bir soket oluştur ve onun tanıtıcı değerini geri döndür
SocketClose	Bir soketi kapat
SocketConnect	Zaman aşımı kontrolü ile sunucuya bağlan
SocketIsConnected	Soketin şu anda bağlı olup olmadığını kontrol et

Fonksiyon	Eylem
SocketIsReadable	Bir soketten okunabilen bir dizi baytı al
SocketIsWritable	Verilerin şu anda bir sokete yazılıp yazılmayacağını kontrol et
SocketTimeouts	Soket sistem nesnesi için veri almak ve göndermek için zaman aşımını ayarla
SocketRead	Bir soketten veri oku
SocketSend	Bir sokete veri yaz
SocketTlsHandshake	TLS Tokalaşma protokolü aracılığıyla belirli bir ana bilgisayara güvenli TLS (SSL) bağlantısı kur
SocketTlsCertificate	Ağ bağlantısını güvenli hale getirmek için kullanılan sertifika hakkında veri al
SocketTlsRead	Verileri güvenli TLS bağlantısından oku
SocketTlsReadAvailable	Kullanılabilir tüm verileri güvenli TLS bağlantısından oku
SocketTlsSend	Verileri güvenli TLS bağlantısıyla gönder
WebRequest	Belirtilen bir sunucuya bir HTTP isteği gönder
SendFTP	FTP sekmesinde belirtilen adrese bir dosya gönder
SendMail	Seçenekler penceresinin E-posta sekmesinde belirtilen adrese bir e-posta gönder
SendNotification	Bildirimler sekmesinde MetaQuotes ID'leri belirtilen mobil terminallere anlık bildirimler gönder

SocketCreate

Belirtilen bayraklara sahip bir soket oluşturur ve onun tanıtıcı değerini geri döndürür.

```
int  SocketCreate(
   uint     flags         // bayraklar
   );
```

Parametreler

flags

[in] Bir soketle çalışma modunu tanımlayan bayrakların kombinasyonu. Şu anda, yalnızca bir bayrak desteklenmektedir - SOCKET_DEFAULT.

Geri dönüş değeri

Başarılı bir soket oluşturulması durumunda tanıtıcı değeri, aksi takdirde INVALID_HANDLE değeri geri döner.

Not

Kullanılmayan bir soketten bilgisayar belleğini boşaltmak için SocketClose'u çağırın.

Bir MQL5 programından maksimum 128 soket oluşturabilirsiniz. Sınır aşılırsa; 5271 hatası (ERR_NETSOCKET_TOO_MANY_OPENED), _LastError'a yazılır.

Fonksiyon, yalnızca kendi yürütme iş parçacıklarında çalışan Uzman Danışmanlardan ve komut dosyalarından çağrılabilir. Bir göstergeden çağrılırsa; GetLastError(), 4014 hatasını geri döndürür - "Çağırma için fonksiyona izin verilmiyor".

Örnek:

```
//+------------------------------------------------------------------+
//|                                                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 "Örneğin çalışmasına izin vermek için terminal ayarlarında izin
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Sunucuya komut gönder                                            |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
```

```
   if(len<0)
      return(false);
//--- 443 numaralı bağlantı noktası üzerinden güvenli TLS bağlantısı kullanılıyorsa
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- standart TCP bağlantısı kullanılıyorsa
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Sunucu yanıtını oku                                              |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- soketten verileri zaman aşımından uzun olmayacak şekilde mevcut oldukları sürece
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- bağlantının güvenli olup olmamasına bağlı olarak çeşitli okuma komutla
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- cevabı analiz et
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- yalnızca yanıt başlığını yazdır
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP cevap başlığı alındı:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
```

```
void OnStart()
  {
   int socket=SocketCreate();
//--- tanıtıcı değerini kontrol et
   if(socket!=INVALID_HANDLE)
     {
      //--- her şey yolundaysa bağlan
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Bağlantı kuruldu: ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- bağlantı sertifika ile güvenliyse, verilerini görüntüle
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS sertifikası:");
            Print("   Sahip:  ",subject);
            Print("   Veren:  ",issuer);
            Print("   Numara:     ",serial);
            Print("   Parmak izi: ",thumbprint);
            Print("   Bitiş tarihi: ",expiration);
            ExtTLS=true;
           }
         //--- sunucuya GET isteği gönder
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET isteği gönderildi");
            //--- cevabı oku
            if(!HTTPRecv(socket,1000))
               Print("Bir yanıt alınamadı, hata ",GetLastError());
           }
         else
            Print("GET isteği gönderilemedi, hata ",GetLastError());
        }
      else
        {
         Print("Bağlantı ",Address,":",Port," kurulamadı, hata ",GetLastError());
        }
      //--- kullandıktan sonra soketi kapat
      SocketClose(socket);
     }
   else
      Print("Soket oluşturulamadı, hata ",GetLastError());
  }
//+------------------------------------------------------------------+
```

SocketClose

Bir soketi kapatır.

```
bool  SocketClose(
   const int  socket      // soket tanıtıcı değeri
   );
```

Parametreler

socket

[in] Kapatılacak soketin tanıtıcı değeri. Tanıtıcı değeri, SocketCreate fonksiyonu tarafından geri döndürülür. Yanlış bir tanıtıcı değeri iletildiğinde; hata 5270 (ERR_NETSOCKET_INVALIDHANDLE), _LastError'a yazılır.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner.

Not

Eğer bir soket için daha önce SocketConnect aracılığıyla bir bağlantı kurulduysa, bu bağlantı kesilir.

Fonksiyon, yalnızca kendi yürütme iş parçacıklarında çalışan Uzman Danışmanlardan ve komut dosyalarından çağrılabilir. Bir göstergeden çağrılırsa; GetLastError(), 4014 hatasını geri döndürür - "Çağırma için fonksiyona izin verilmiyor".

Örnek:

```
//+------------------------------------------------------------------+
//|                                                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 "Örneğin çalışmasına izin vermek için terminal ayarlarında izin
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Sunucuya komut gönder                                            |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
```

```
//--- 443 numaralı bağlantı noktası üzerinden güvenli TLS bağlantısı kullanılıyorsa
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- standart TCP bağlantısı kullanılıyorsa
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Sunucu yanıtını oku                                              |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- soketten verileri zaman aşımından uzun olmayacak şekilde mevcut oldukları sürece
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- bağlantının güvenli olup olmamasına bağlı olarak çeşitli okuma komutla
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- cevabı analiz et
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- yalnızca yanıt başlığını yazdır
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP cevap başlığı alındı:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
```

```
   int socket=SocketCreate();
//--- tanıtıcı değerini kontrol et
   if(socket!=INVALID_HANDLE)
     {
      //--- her şey yolundaysa bağlan
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Bağlantı kuruldu: ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- bağlantı sertifika ile güvenliyse, verilerini görüntüle
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS sertifikası:");
            Print("   Sahip:  ",subject);
            Print("   Veren:  ",issuer);
            Print("   Numara:     ",serial);
            Print("   Parmak izi: ",thumbprint);
            Print("   Bitiş tarihi: ",expiration);
            ExtTLS=true;
           }
         //--- sunucuya GET isteği gönder
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET isteği gönderildi");
            //--- cevabı oku
            if(!HTTPRecv(socket,1000))
               Print("Bir yanıt alınamadı, hata ",GetLastError());
           }
         else
            Print("GET isteği gönderilemedi, hata ",GetLastError());
        }
      else
        {
         Print("Bağlantı ",Address,":",Port," kurulamadı, hata ",GetLastError());
        }
      //--- kullandıktan sonra soketi kapat
      SocketClose(socket);
     }
   else
      Print("Soket oluşturulamadı, hata ",GetLastError());
  }
//+------------------------------------------------------------------+
```

SocketConnect

Zaman aşımı kontrolü ile sunucuya bağlanır.

```
bool  SocketConnect(
   int           socket,              // soket
   const string  server,              // bağlantı adresi
   uint          port,                // bağlantı noktası
   uint          timeout_receive_ms   // bağlantı zaman aşımı
   );
```

Parametreler

socket

[in] SocketCreate fonksiyonu tarafından geri döndürülen soket tanıtıcı değeri. Yanlış bir tanıtıcı değeri iletildiğinde; hata 5270 (ERR_NETSOCKET_INVALIDHANDLE), _LastError'a yazılır.

server

[in] Bağlanmak istediğiniz sunucunun alan adı veya IP adresi.

port

[in] Bağlantı noktası numarası.

timeout_receive_ms

[in] Milisaniye cinsinden bağlantı zaman aşımı. Bu süre içerisinde bağlantı kurulmazsa, bağlantı denemeleri durdurulur.

Geri dönüş değeri

Eğer bağlantı başarılı olursa true, aksi takdirde false olarak geri döner.

Not

Bağlantı adresi, müşteri terminali tarafındaki izin verilenler listesine eklenmelidir (Araçlar \ Seçenekler \ Uzman Danışmanlar).

Bağlantı başarısız olursa; hata 5272 (ERR_NETSOCKET_CANNOT_CONNECT), _LastError'a yazılır.

Fonksiyon, yalnızca kendi yürütme iş parçacıklarında çalışan Uzman Danışmanlardan ve komut dosyalarından çağrılabilir. Bir göstergeden çağrılırsa; GetLastError(), 4014 hatasını geri döndürür - "Çağırma için fonksiyona izin verilmiyor".

Örnek:

```
//+------------------------------------------------------------------+
//|                                                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 "Örneğin çalışmasına izin vermek için terminal ayarlarında izin
#property script_show_inputs
```

```
input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Sunucuya komut gönder                                            |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 443 numaralı bağlantı noktası üzerinden güvenli TLS bağlantısı kullanılıyorsa
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- standart TCP bağlantısı kullanılıyorsa
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Sunucu yanıtını oku                                              |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- soketten verileri zaman aşımından uzun olmayacak şekilde mevcut oldukları sürece
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- bağlantının güvenli olup olmamasına bağlı olarak çeşitli okuma komutlar
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- cevabı analiz et
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- yalnızca yanıt başlığını yazdır
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP cevap başlığı alındı:");
               Print(StringSubstr(result,0,header_end));
```

```
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- tanıtıcı değerini kontrol et
   if(socket!=INVALID_HANDLE)
     {
      //--- her şey yolundaysa bağlan
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Bağlantı kuruldu: ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- bağlantı sertifika ile güvenliyse, verilerini görüntüle
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS sertifikası:");
            Print("   Sahip:  ",subject);
            Print("   Veren:  ",issuer);
            Print("   Numara:     ",serial);
            Print("   Parmak izi: ",thumbprint);
            Print("   Bitiş tarihi: ",expiration);
            ExtTLS=true;
           }
         //--- sunucuya GET isteği gönder
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET isteği gönderildi");
            //--- cevabı oku
            if(!HTTPRecv(socket,1000))
               Print("Bir yanıt alınamadı, hata ",GetLastError());
           }
         else
            Print("GET isteği gönderilemedi, hata ",GetLastError());
        }
      else
        {
         Print("Bağlantı ",Address,":",Port," kurulamadı, hata ",GetLastError());
```

```
         }
       //--- kullandıktan sonra soketi kapat
       SocketClose(socket);
      }
    else
       Print("Soket oluşturulamadı, hata ",GetLastError());
  }
//+------------------------------------------------------------------+
```

SocketIsConnected

Soketin şu anda bağlı olup olmadığını kontrol eder.

```
bool  SocketIsConnected(
   const int  socket      // soket tanıtıcı değeri
   );
```

Parametreler

socket

[in] SocketCreate() fonksiyonu tarafından geri döndürülen soket tanıtıcı değeri. _LastError'a yanlış bir tanıtıcı değeri iletildiğinde, 5270 hatası (ERR_NETSOCKET_INVALIDHANDLE) etkinleştirilir.

Geri dönüş değeri

Soket bağlıysa true, aksi halde false olarak geri döner.

Not

SocketIsConnected() fonksiyonu mevcut soket bağlantı durumunun kontrol edilmesini sağlar.

Fonksiyon, yalnızca kendi yürütme iş parçacıklarında çalışan Uzman Danışmanlardan ve komut dosyalarından çağrılabilir. Bir göstergeden çağrılırsa; GetLastError(), 4014 hatasını geri döndürür - "Çağırma için fonksiyona izin verilmiyor".

Ayrıca bakınız

SocketConnect, SocketIsWritable, SocketCreate, SocketClose

SocketIsReadable

Bir soketten okunabilen bir dizi baytı alır.

```
uint  SocketIsReadable(
   const int  socket      // soket tanıtıcı değeri
   );
```

Parametreler

socket

[in] SocketCreate fonksiyonu tarafından geri döndürülen soket tanıtıcı değeri. _LastError'a yanlış bir tanıtıcı değeri iletildiğinde, 5270 hatası (ERR_NETSOCKET_INVALIDHANDLE) etkinleştirilir.

Geri dönüş değeri

Okunabilen bayt sayısı. Bir hata durumunda, 0 geri döner.

Not

Fonksiyon yürütülürken bir sistem soketinde bir hata oluşursa, SocketConnect aracılığıyla kurulan bağlantı kesilir.

SocketRead'i çağırmadan önce, sokette okumak için veri olup olmadığını kontrol edin. Aksi takdirde, veri yoksa, SocketRead fonksiyonu programın yürütümünü geciktiren timeout_ms süresi boyunca verileri bekler.

Fonksiyon, yalnızca kendi yürütme iş parçacıklarında çalışan Uzman Danışmanlardan ve komut dosyalarından çağrılabilir. Bir göstergeden çağrılırsa; GetLastError(), 4014 hatasını geri döndürür - "Çağırma için fonksiyona izin verilmiyor".

Örnek:

```
//+------------------------------------------------------------------+
//|                                                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 "Örneğin çalışmasına izin vermek için terminal ayarlarında izin
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Sunucuya komut gönder                                            |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
```

```
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 443 numaralı bağlantı noktası üzerinden güvenli TLS bağlantısı kullanılıyorsa
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- standart TCP bağlantısı kullanılıyorsa
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Sunucu yanıtını oku                                              |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- soketten verileri zaman aşımından uzun olmayacak şekilde mevcut oldukları sürece
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- bağlantının güvenli olup olmamasına bağlı olarak çeşitli okuma komutları
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- cevabı analiz et
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- yalnızca yanıt başlığını yazdır
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP cevap başlığı alındı:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
```

```
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- tanıtıcı değerini kontrol et
   if(socket!=INVALID_HANDLE)
     {
      //--- her şey yolundaysa bağlan
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Bağlantı kuruldu: ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- bağlantı sertifika ile güvenliyse, verilerini görüntüle
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS sertifikası:");
            Print("   Sahip:  ",subject);
            Print("   Veren:  ",issuer);
            Print("   Numara:     ",serial);
            Print("   Parmak izi: ",thumbprint);
            Print("   Bitiş tarihi: ",expiration);
            ExtTLS=true;
           }
         //--- sunucuya GET isteği gönder
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET isteği gönderildi");
            //--- cevabı oku
            if(!HTTPRecv(socket,1000))
               Print("Bir yanıt alınamadı, hata ",GetLastError());
           }
         else
            Print("GET isteği gönderilemedi, hata ",GetLastError());
        }
      else
        {
         Print("Bağlantı ",Address,":",Port," kurulamadı, hata ",GetLastError());
        }
      //--- kullandıktan sonra soketi kapat
      SocketClose(socket);
     }
   else
      Print("Soket oluşturulamadı, hata ",GetLastError());
  }
//+------------------------------------------------------------------+
```

SocketIsWritable

Verilerin şu anda bir sokete yazılıp yazılmayacağını kontrol eder.

```
bool  SocketIsWritable(
   const int  socket      // soket tanıtıcı değeri
   );
```

Parametreler

socket

[in] SocketCreate fonksiyonu tarafından geri döndürülen soket tanıtıcı değeri. Yanlış bir tanıtıcı değeri iletildiğinde; hata 5270 (ERR_NETSOCKET_INVALIDHANDLE), _LastError'a yazılır.

Geri dönüş değeri

Yazma mümkünse true, aksi halde false olarak geri döner.

Not

Bu fonksiyon, şu anda bir sokete veri yazmanın mümkün olup olmadığını kontrol etmenizi sağlar.

Fonksiyon yürütülürken bir sistem soketinde bir hata oluşursa, SocketConnect aracılığıyla kurulan bağlantı kesilir.

Fonksiyon, yalnızca kendi yürütme iş parçacıklarında çalışan Uzman Danışmanlardan ve komut dosyalarından çağrılabilir. Bir göstergeden çağrılırsa; GetLastError(), 4014 hatasını geri döndürür - "Çağırma için fonksiyona izin verilmiyor".

SocketTimeouts

Soket sistem nesnesi için veri alır ve göndermek için zaman aşımını ayarlar.

```
bool  SocketTimeouts(
   int           socket,               // soket
   uint          timeout_send_ms,      // veri gönderme zaman aşımı
   uint          timeout_receive_ms    // veri alma zaman aşımı
   );
```

Parametreler

socket

[in] SocketCreate fonksiyonu tarafından geri döndürülen soket tanıtıcı değeri. Yanlış bir tanıtıcı değeri iletildiğinde; hata 5270 (ERR_NETSOCKET_INVALIDHANDLE), _LastError'a yazılır.

timeout_send_ms

[in] Milisaniye cinsinden veri gönderme zaman aşımı.

timeout_receive_ms

[in] Milisaniye cinsinden veri alma zaman aşımı.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner.

Not

SocketRead üzerinden veri okurken sistem nesnelerinin zaman aşımını ve ayarlanmış zaman aşımlarını karıştırmayın. SocketTimeout, işletim sistemindeki bir soket nesnesi için zaman aşımını bir kez ayarlar. Bu zaman aşımları, bu soket üzerinden veri okuma ve gönderme fonksiyonlarının tümüne uygulanacaktır. SocketRead'de zaman aşımı, belirli bir veri okuma işlemi için ayarlanmıştır.

Fonksiyon, yalnızca kendi yürütme iş parçacıklarında çalışan Uzman Danışmanlardan ve komut dosyalarından çağrılabilir. Bir göstergeden çağrılırsa; GetLastError(), 4014 hatasını geri döndürür - "Çağırma için fonksiyona izin verilmiyor".

SocketRead

Bir soketten veri okur.

```
int  SocketRead(
   int           socket,              // soket
   uchar&        buffer[],            // soketten veri okumak için tampon
   uint          buffer_maxlen,       // okunacak bayt sayısı
   uint          timeout_ms           // okuma zaman aşımı
   );
```

Parametreler

socket

[in] SocketCreate fonksiyonu tarafından geri döndürülen soket tanıtıcı değeri. _LastError'a yanlış bir tanıtıcı değeri iletildiğinde, 5270 hatası (ERR_NETSOCKET_INVALIDHANDLE) etkinleştirilir.

buffer

[out] Veri okunan uchar dizisine referans. Dinamik dizi boyutu, okunan bayt sayısıyla artar. Dizi boyutu, INT_MAX (2147483647) değerini aşamaz.

buffer_maxlen

[in] *buffer[]* dizisine okunacak bayt sayısı. Diziye sığmayan veriler sokette kalır. Bu veriler bir sonraki SocketRead çağrısı tarafından alınabilirler. *buffer_maxlen*, INT_MAX (2147483647) değerini aşamaz.

timeout_ms

[in] Milisaniye cinsinden veri okuma zaman aşımı. Bu süre içinde veri alınmazsa, okuma denemeleri durdurulur ve fonksiyon -1 geri döndürür.

Geri dönüş değeri

Başarılı olursa, okuma baytlarının sayısı geri döner. Bir hata durumunda, 0 geri döner.

Not

Fonksiyon yürütülürken bir sistem soketinde bir hata oluşursa, SocketConnect aracılığıyla kurulan bağlantı kesilir.

Veri okuma hatası olması durumunda; hata 5273 (ERR_NETSOCKET_IO_ERROR), _LastError'a yazılır.

Fonksiyon, yalnızca kendi yürütme iş parçacıklarında çalışan Uzman Danışmanlardan ve komut dosyalarından çağrılabilir. Bir göstergeden çağrılırsa; GetLastError(), 4014 hatasını geri döndürür - "Çağırma için fonksiyona izin verilmiyor".

Örnek:

```
//+------------------------------------------------------------------+
//|                                                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 "Örneğin çalışmasına izin vermek için terminal ayarlarında izin
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Sunucuya komut gönder                                            |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 443 numaralı bağlantı noktası üzerinden güvenli TLS bağlantısı kullanılıyorsa
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- standart TCP bağlantısı kullanılıyorsa
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Sunucu yanıtını oku                                              |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- soketten verileri zaman aşımından uzun olmayacak şekilde mevcut oldukları sürece
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- bağlantının güvenli olup olmamasına bağlı olarak çeşitli okuma komutla
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- cevabı analiz et
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- yalnızca yanıt başlığını yazdır
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
```

```
               {
                Print("HTTP cevap başlığı alındı:");
                Print(StringSubstr(result,0,header_end));
                return(true);
               }
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- tanıtıcı değerini kontrol et
   if(socket!=INVALID_HANDLE)
     {
      //--- her şey yolundaysa bağlan
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Bağlantı kuruldu: ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- bağlantı sertifika ile güvenliyse, verilerini görüntüle
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS sertifikası:");
            Print("   Sahip:  ",subject);
            Print("   Veren:  ",issuer);
            Print("   Numara:     ",serial);
            Print("   Parmak izi: ",thumbprint);
            Print("   Bitiş tarihi: ",expiration);
            ExtTLS=true;
           }
         //--- sunucuya GET isteği gönder
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET isteği gönderildi");
            //--- cevabı oku
            if(!HTTPRecv(socket,1000))
               Print("Bir yanıt alınamadı, hata ",GetLastError());
           }
         else
            Print("GET isteği gönderilemedi, hata ",GetLastError());
        }
```

```
      else
        {
         Print("Bağlantı ",Address,":",Port," kurulamadı, hata ",GetLastError());
        }
      //--- kullandıktan sonra soketi kapat
      SocketClose(socket);
     }
   else
      Print("Soket oluşturulamadı, hata ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Ayrıca bakınız

SocketTimeouts, MathSwap

SocketSend

Bir sokete veri yazar.

```
int  SocketSend(
   int           socket,              // soket
   const uchar&  buffer[],            // veri tamponu
   uint          buffer_len           // tampon boyutu
   );
```

Parametreler

socket

[in] SocketCreate fonksiyonu tarafından geri döndürülen soket tanıtıcı değeri. Yanlış bir tanıtıcı değeri iletildiğinde; hata 5270 (ERR_NETSOCKET_INVALIDHANDLE), _LastError'a yazılır.

buffer

[in] Sokete gönderilecek olan verileri içeren uchar dizisine referans.

buffer_len

[in] 'buffer' dizi boyutu.

Geri dönüş değeri

Başarılı olursa, sokete yazılan bayt sayısı geri döner. Bir hata durumunda, 0 geri döner.

Not

Fonksiyon yürütülürken bir sistem soketinde bir hata oluşursa, SocketConnect aracılığıyla kurulan bağlantı kesilir.

Veri yazma hatası olması durumunda; hata 5273 (ERR_NETSOCKET_IO_ERROR), _LastError'a yazılır.

Fonksiyon, yalnızca kendi yürütme iş parçacıklarında çalışan Uzman Danışmanlardan ve komut dosyalarından çağrılabilir. Bir göstergeden çağrılırsa; GetLastError(), 4014 hatasını geri döndürür - "Çağırma için fonksiyona izin verilmiyor".

Örnek:

```
//+------------------------------------------------------------------+
//|                                                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 "Örneğin çalışmasına izin vermek için terminal ayarlarında izin
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
```

```
//+------------------------------------------------------------------+
//| Sunucuya komut gönder                                            |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 443 numaralı bağlantı noktası üzerinden güvenli TLS bağlantısı kullanılıyorsa
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- standart TCP bağlantısı kullanılıyorsa
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Sunucu yanıtını oku                                              |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- soketten verileri zaman aşımından uzun olmayacak şekilde mevcut oldukları sürece
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- bağlantının güvenli olup olmamasına bağlı olarak çeşitli okuma komutla
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- cevabı analiz et
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- yalnızca yanıt başlığını yazdır
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP cevap başlığı alındı:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
```

```
      }
    while(GetTickCount()<timeout_check && !IsStopped());
    return(false);
   }
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- tanıtıcı değerini kontrol et
   if(socket!=INVALID_HANDLE)
     {
      //--- her şey yolundaysa bağlan
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Bağlantı kuruldu: ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- bağlantı sertifika ile güvenliyse, verilerini görüntüle
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS sertifikası:");
            Print("   Sahip:  ",subject);
            Print("   Veren:  ",issuer);
            Print("   Numara:     ",serial);
            Print("   Parmak izi: ",thumbprint);
            Print("   Bitiş tarihi: ",expiration);
            ExtTLS=true;
           }
         //--- sunucuya GET isteği gönder
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET isteği gönderildi");
            //--- cevabı oku
            if(!HTTPRecv(socket,1000))
               Print("Bir yanıt alınamadı, hata ",GetLastError());
           }
         else
            Print("GET isteği gönderilemedi, hata ",GetLastError());
        }
      else
        {
         Print("Bağlantı ",Address,":",Port," kurulamadı, hata ",GetLastError());
        }
      //--- kullandıktan sonra soketi kapat
      SocketClose(socket);
     }
```

```
   else
      Print("Soket oluşturulamadı, hata ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Ayrıca bakınız

SocketTimeouts, MathSwap, StringToCharArray

SocketTlsHandshake

TLS Tokalaşma protokolü aracılığıyla belirli bir ana bilgisayara güvenli TLS (SSL) bağlantısı kurar. Tokalaşma sırasında, bir müşteri ve bir sunucu bağlantı parametreleri üzerinde hemfikir olur: uygulanan protokol sürümü ve veri şifreleme yöntemi.

```
bool  SocketTlsHandshake(
   int           socket,               // soket
   const string  host                  // ana bilgisayar adresi
   );
```

Parametreler

socket

[in] SocketCreate fonksiyonu tarafından geri döndürülen soket tanıtıcı değeri. Yanlış bir tanıtıcı değeri iletildiğinde; hata 5270 (ERR_NETSOCKET_INVALIDHANDLE), _LastError'a yazılır.

host

[in] Güvenli bir bağlantı kurulan sunucunun adresi.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner.

Not

Güvenli bir bağlantıdan önce, program SocketConnect kullanarak ana bilgisayarla standart bir TCP bağlantısı kurmalıdır.

Güvenli bağlantı başarısız olursa; hata 5274 (ERR_NETSOCKET_HANDSHAKE_FAILED), _LastError'a yazılır.

443 numaralı bağlantı noktasına bağlanırken fonksiyonu çağırmanıza gerek yoktur. Bu, güvenli TLS (SSL) bağlantıları için kullanılan standart bir TCP bağlantı noktasıdır.

Fonksiyon, yalnızca kendi yürütme iş parçacıklarında çalışan Uzman Danışmanlardan ve komut dosyalarından çağrılabilir. Bir göstergeden çağrılırsa; GetLastError(), 4014 hatasını geri döndürür - "Çağırma için fonksiyona izin verilmiyor".

SocketTlsCertificate

Ağ bağlantısını güvenli hale getirmek için kullanılan sertifika hakkında veri alır.

```
int  SocketTlsCertificate(
   int           socket,              // soket
   string&       subject,             // sertifika sahibi
   string&       issuer,              // sertifikayı veren
   string&       serial,              // sertifika seri numarası
   string&       thumbprint,          // sertifika parmak izi
   datetime&     expiration           // sertifika bitiş tarihi
   );
```

Parametreler

socket

[in] SocketCreate fonksiyonu tarafından geri döndürülen soket tanıtıcı değeri. Yanlış bir tanıtıcı değeri iletildiğinde; hata 5270 (ERR_NETSOCKET_INVALIDHANDLE), _LastError'a yazılır.

subject

[in] Sertifika sahibinin adı. Subject alanına karşılık gelir.

issuer

[in] Sertifika verenin adı. Issuer alanına karşılık gelir.

serial

[in] Sertifika seri numarası. SerialNumber alanına karşılık gelir.

thumbprint

[in] Sertifika parmak izi. Sertifika dosyasının tamamındaki SHA-1 hash değerine karşılık gelir (verenin imzası dahil tüm alanlar).

expiration

[in] datetime biçimindeki sertifika bitiş tarihi.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner.

Not

Sertifika verileri yalnızca SocketTlsHandshake kullanılarak güvenli bir bağlantı kurulduktan sonra talep edilebilir.

Sertifika elde etme hatası durumunda; hata 5275 (ERR_NETSOCKET_NO_CERTIFICATE), _LastError'a yazılır.

Fonksiyon, yalnızca kendi yürütme iş parçacıklarında çalışan Uzman Danışmanlardan ve komut dosyalarından çağrılabilir. Bir göstergeden çağrılırsa; GetLastError(), 4014 hatasını geri döndürür - "Çağırma için fonksiyona izin verilmiyor".

Örnek:

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

```
//|                                                  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 "Örneğin çalışmasına izin vermek için terminal ayarlarında izin
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Sunucuya komut gönder                                            |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 443 numaralı bağlantı noktası üzerinden güvenli TLS bağlantısı kullanılıyorsa
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- standart TCP bağlantısı kullanılıyorsa
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Sunucu yanıtını oku                                              |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- soketten verileri zaman aşımından uzun olmayacak şekilde mevcut oldukları sürece
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- bağlantının güvenli olup olmamasına bağlı olarak çeşitli okuma komutla
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- cevabı analiz et
```

```
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- yalnızca yanıt başlığını yazdır
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP cevap başlığı alındı:");
               Print(StringSubstr(result,0,header_end));
               return(true);
              }
           }
        }
     }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- tanıtıcı değerini kontrol et
   if(socket!=INVALID_HANDLE)
     {
      //--- her şey yolundaysa bağlan
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Bağlantı kuruldu: ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- bağlantı sertifika ile güvenliyse, verilerini görüntüle
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS sertifikası:");
            Print("   Sahip:  ",subject);
            Print("   Veren:  ",issuer);
            Print("   Numara:     ",serial);
            Print("   Parmak izi: ",thumbprint);
            Print("   Bitiş tarihi: ",expiration);
            ExtTLS=true;
           }
         //--- sunucuya GET isteği gönder
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET isteği gönderildi");
            //--- cevabı oku
```

```
            if(!HTTPRecv(socket,1000))
               Print("Bir yanıt alınamadı, hata ",GetLastError());
            }
         else
            Print("GET isteği gönderilemedi, hata ",GetLastError());
        }
      else
        {
         Print("Bağlantı ",Address,":",Port," kurulamadı, hata ",GetLastError());
        }
      //--- kullandıktan sonra soketi kapat
      SocketClose(socket);
     }
   else
      Print("Soket oluşturulamadı, hata ",GetLastError());
  }
//+------------------------------------------------------------------+
```

SocketTlsRead

Verileri güvenli TLS bağlantısından okur.

```
int  SocketTlsRead(
   int           socket,              // soket
   uchar&        buffer[],            // soketten veri okumak için tampon
   uint          buffer_maxlen        // okunacak bayt sayısı
   );
```

Parametreler

socket

[in] SocketCreate fonksiyonu tarafından geri döndürülen soket tanıtıcı değeri. _LastError'a yanlış bir tanıtıcı değeri iletildiğinde, 5270 hatası (ERR_NETSOCKET_INVALIDHANDLE) etkinleştirilir.

buffer

[out] Veri okunan uchar dizisine referans. Dinamik dizi boyutu, okunan bayt sayısıyla artar. Dizi boyutu, INT_MAX (2147483647) değerini aşamaz.

buffer_maxlen

[in] *buffer[]* dizisine okunacak bayt sayısı. Diziye sığmayan veriler sokette kalır. Bu veriler bir sonraki SocketTLSRead çağrısı tarafından alınabilirler. *buffer_maxlen*, INT_MAX (2147483647) değerini aşamaz.

Geri dönüş değeri

Başarılı olursa, okuma baytlarının sayısı geri döner. Bir hata durumunda, 0 geri döner.

Not

Fonksiyon yürütülürken bir sistem soketinde bir hata oluşursa, SocketConnect aracılığıyla kurulan bağlantı kesilir.

Fonksiyon, belirtilen miktarda veri alana kadar veya zaman aşımına ulaşana kadar yürütülür (SocketTimeouts).

Veri okuma hatası olması durumunda; hata 5273 (ERR_NETSOCKET_IO_ERROR), _LastError'a yazılır.

Fonksiyon, yalnızca kendi yürütme iş parçacıklarında çalışan Uzman Danışmanlardan ve komut dosyalarından çağrılabilir. Bir göstergeden çağrılırsa; GetLastError(), 4014 hatasını geri döndürür - "Çağırma için fonksiyona izin verilmiyor".

Örnek:

```
//+------------------------------------------------------------------+
//|                                                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 "Örneğin çalışmasına izin vermek için terminal ayarlarında izin
```

```
#property script_show_inputs

input string Address="www.mql5.com";
input int    Port   =80;
bool         ExtTLS =false;
//+------------------------------------------------------------------+
//| Sunucuya komut gönder                                            |
//+------------------------------------------------------------------+
bool HTTPSend(int socket,string request)
  {
   char req[];
   int  len=StringToCharArray(request,req)-1;
   if(len<0)
      return(false);
//--- 443 numaralı bağlantı noktası üzerinden güvenli TLS bağlantısı kullanılıyorsa
   if(ExtTLS)
      return(SocketTlsSend(socket,req,len)==len);
//--- standart TCP bağlantısı kullanılıyorsa
   return(SocketSend(socket,req,len)==len);
  }
//+------------------------------------------------------------------+
//| Sunucu yanıtını oku                                              |
//+------------------------------------------------------------------+
bool HTTPRecv(int socket,uint timeout)
  {
   char   rsp[];
   string result;
   uint   timeout_check=GetTickCount()+timeout;
//--- soketten verileri zaman aşımından uzun olmayacak şekilde mevcut oldukları sürece
   do
     {
      uint len=SocketIsReadable(socket);
      if(len)
        {
         int rsp_len;
         //--- bağlantının güvenli olup olmamasına bağlı olarak çeşitli okuma komutla
         if(ExtTLS)
            rsp_len=SocketTlsRead(socket,rsp,len);
         else
            rsp_len=SocketRead(socket,rsp,len,timeout);
         //--- cevabı analiz et
         if(rsp_len>0)
           {
            result+=CharArrayToString(rsp,0,rsp_len);
            //--- yalnızca yanıt başlığını yazdır
            int header_end=StringFind(result,"\r\n\r\n");
            if(header_end>0)
              {
               Print("HTTP cevap başlığı alındı:");
```

```
                Print(StringSubstr(result,0,header_end));
                return(true);
               }
            }
         }
      }
   while(GetTickCount()<timeout_check && !IsStopped());
   return(false);
  }
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   int socket=SocketCreate();
//--- tanıtıcı değerini kontrol et
   if(socket!=INVALID_HANDLE)
     {
      //--- her şey yolundaysa bağlan
      if(SocketConnect(socket,Address,Port,1000))
        {
         Print("Bağlantı kuruldu: ",Address,":",Port);

         string   subject,issuer,serial,thumbprint;
         datetime expiration;
         //--- bağlantı sertifika ile güvenliyse, verilerini görüntüle
         if(SocketTlsCertificate(socket,subject,issuer,serial,thumbprint,expiration))
           {
            Print("TLS sertifikası:");
            Print("   Sahip:  ",subject);
            Print("   Veren:  ",issuer);
            Print("   Numara:     ",serial);
            Print("   Parmak izi: ",thumbprint);
            Print("   Bitiş tarihi: ",expiration);
            ExtTLS=true;
           }
         //--- sunucuya GET isteği gönder
         if(HTTPSend(socket,"GET / HTTP/1.1\r\nHost: www.mql5.com\r\nUser-Agent: MT5\r
           {
            Print("GET isteği gönderildi");
            //--- cevabı oku
            if(!HTTPRecv(socket,1000))
               Print("Bir yanıt alınamadı, hata ",GetLastError());
           }
         else
            Print("GET isteği gönderilemedi, hata ",GetLastError());
        }
      else
        {
```

```
         Print("Bağlantı ",Address,":",Port," kurulamadı, hata ",GetLastError());
        }
      //--- kullandıktan sonra soketi kapat
      SocketClose(socket);
     }
   else
      Print("Soket oluşturulamadı, hata ",GetLastError());
  }
//+------------------------------------------------------------------+
```

Ayrıca bakınız

SocketTimeouts, MathSwap

SocketTlsReadAvailable

Kullanılabilir tüm verileri güvenli TLS bağlantısından okur.

```
int  SocketTlsReadAvailable(
   int           socket,              // soket
   uchar&        buffer[],            // soketten veri okumak için tampon
   const uint    buffer_maxlen        // okunacak bayt sayısı
   );
```

Parametreler

socket

[in] SocketCreate fonksiyonu tarafından geri döndürülen soket tanıtıcı değeri. _LastError'a yanlış bir tanıtıcı değeri iletildiğinde, 5270 hatası (ERR_NETSOCKET_INVALIDHANDLE) etkinleştirilir.

buffer

[out] Veri okunan uchar dizisine referans. Dinamik dizi boyutu, okunan bayt sayısıyla artar. Dizi boyutu, INT_MAX (2147483647) değerini aşamaz.

buffer_maxlen

[in] buffer[] dizisine okunacak bayt sayısı. Diziye sığmayan veriler sokette kalır. Bu veriler bir sonraki SocketTlsReadAvailable veya SocketTlsRead çağrısı tarafından alınabilirler. buffer_maxlen, INT_MAX (2147483647) değerini aşamaz.

Geri dönüş değeri

Başarılı olursa, okuma baytlarının sayısı geri döner. Bir hata durumunda, 0 geri döner.

Not

Fonksiyon yürütülürken bir sistem soketinde bir hata oluşursa, SocketConnect aracılığıyla kurulan bağlantı kesilir.

Veri okuma hatası olması durumunda; hata 5273 (ERR_NETSOCKET_IO_ERROR), _LastError'a yazılır.

Fonksiyon, yalnızca kendi yürütme iş parçacıklarında çalışan Uzman Danışmanlardan ve komut dosyalarından çağrılabilir. Bir göstergeden çağrılırsa; GetLastError(), 4014 hatasını geri döndürür - "Çağırma için fonksiyona izin verilmiyor".

Ayrıca bakınız

SocketTimeouts, MathSwap

SocketTlsSend

Verileri güvenli TLS bağlantısıyla gönderir.

```
int  SocketTlsSend(
   int           socket,              // soket
   const uchar&  buffer[],            // veri tamponu
   uint          buffer_len           // tampon boyutu
   );
```

Parametreler

socket

[in] SocketCreate fonksiyonu tarafından geri döndürülen soket tanıtıcı değeri. Yanlış bir tanıtıcı değeri iletildiğinde; hata 5270 (ERR_NETSOCKET_INVALIDHANDLE), _LastError'a yazılır.

buffer

[in] Gönderilecek olan verileri içeren uchar dizisine referans.

buffer_len

[in] 'buffer' dizi boyutu.

Geri dönüş değeri

Başarılı olursa, sokete yazılan bayt sayısı geri döner. Bir hata durumunda, 0 geri döner.

Not

Fonksiyon yürütülürken bir sistem soketinde bir hata oluşursa, SocketConnect aracılığıyla kurulan bağlantı kesilir.

Veri yazma hatası olması durumunda; hata 5273 (ERR_NETSOCKET_IO_ERROR), _LastError'a yazılır.

Fonksiyon, yalnızca kendi yürütme iş parçacıklarında çalışan Uzman Danışmanlardan ve komut dosyalarından çağrılabilir. Bir göstergeden çağrılırsa; GetLastError(), 4014 hatasını geri döndürür - "Çağırma için fonksiyona izin verilmiyor".

Ayrıca bakınız

SocketTimeouts, MathSwap, StringToCharArray

WebRequest

Fonksiyon, belirtilen bir sunucuya bir HTTP isteği gönderir. Fonksiyon iki versiyona sahiptir:

1. **Basit istekler** "key=value" türünde gönderme Content-Type: application/x-www-form-urlencoded başlığını kullanarak.

```
int  WebRequest(
   const string      method,           // HTTP metodu
   const string      url,              // URLbaşlıklar
   const string      cookie,           // çerez
   const string      referer,          // başvurucu
   int               timeout,          // zaman aşımı
   const char        &data[],          // HTTP mesaj gövdesinin dizisi
   int               data_size,        // data[] dizi boyutu bayt cinsinden
   char              &result[],        // sunucu yanıt verisini içeren bir dizi
   string            &result_headers   // sunucu yanıt başlıkları
   );
```

2. **Çeşitli Web hizmetleriyle daha esnek bir etkileşim için** özel başlık kümelerini belirten herhangi bir türden istek göndermek.

```
int  WebRequest(
   const string      method,           // HTTP metodu
   const string      url,              // URLbaşlıklar
   const string      headers,          // başlıklar
   int               timeout,          // zaman aşımı
   const char        &data[],          // HTTP mesaj gövdesinin dizisi
   char              &result[],        // sunucu yanıt verisini içeren bir dizi
   string            &result_headers   // sunucu yanıt başlıkları
   );
```

Parametreler

method

[in] HTTP metodu.

url

[in] URL.

headers

[in] Request headers of type "key: value" türünde başlıklar isteği, "\r\n" satır sonu ile ayrılmış.

cookie

[in] Çerez değeri.

referer

[in] HTTP isteğinin Başvurucu başlığının değeri

timeout

[in] Milisaniye cinsinden zaman aşımı.

data[]

[in] HTTP mesaj gövdesinin veri dizisi.

data_size

[in] daata[] dizininin boyutu.

result[]

[out] Sunucu yanıt verisini içeren bir dizi.

result_headers

[out] Sunucu cevap başlıkları.

Dönüş değeri

HTTP sunucusu yanıt kodu veya bir hata için -1.

Not

WebRequest () işlevini kullanmak için, gerekli sunucuların adreslerini "Seçenekler" penceresinin "Uzman Danışmanlar" sekmesinde izin verilen URL'ler listesine ekleyin. Sunucu portu, belirtilen protokol temelinde otomatik olarak seçilir - "http://" için 80 ve "https://" için 443.

WebRequest() onksiyonu senkronize edilir, yani programın çalışmasını bozar ve istenen sunucudan yanıt bekler. Bir yanıt almadaki gecikmeler büyük olabileceğinden, fonksiyonlar göstergelerden gelen çağrılar için mevcut değildir, çünkü göstergeler tüm semboller ve grafikler tarafından tek bir sembolde paylaşılan ortak bir iş parçacığında çalışır. Bir sembolün grafiklerinden birinde göstergenin performans gecikmesi, aynı sembolün tüm grafiklerinin güncellenmesini durdurabilir.

Fonksiyon, kendi yürütme dizilerinde çalıştıkları için yalnızca Uzman Danışmanlar ve komut dosyalarından çağrılabilir. Bir göstergeden fonksiyonu çağırmaya çalışırsanız, GetLastError() Hata 4014 dönecektir- "Fonksiyona izin verilmiyor".

WebRequest(), Strateji Testerda yürütülemez.

Örnek:

```
void OnStart()
  {
   string cookie=NULL,headers;
   char   post[],result[];
   string url="https://finance.yahoo.com";
//--- Sunucuya erişimi etkinleştirmek için "https://finance.yahoo.com" URL'si eklemeli
//--- izin verilen URL'ler listesine (Ana Menü->Araçlar->Seçenekler, "Uzman Danışmanla
//--- Son hata kodunu sıfırlama
   ResetLastError();
//--- Yahoo Finans'dan bir html sayfası indirme
   int res=WebRequest("GET",url,cookie,NULL,500,post,0,result,headers);
   if(res==-1)
     {
      Print("Error in WebRequest. Error code  =",GetLastError());
      //--- Belki de URL listelenmemiş, adresi eklemenin gerekliliği hakkında bir mesa
      MessageBox("Adresi ekleyin '"+url+"' 'Uzman Danışmanlar' sekmesindeki izin veril
     }
```

```
   else
     {
      if(res==200)
        {
         //--- İndirme başarılı
         PrintFormat("Dosya başarılı bir şekilde indirildi, Dosya boyutu %d byte.",Ar
         //PrintFormat("Server headers: %s",headers);
         //--- Datayı bir doyaya kaydetme
         int filehandle=FileOpen("url.htm",FILE_WRITE|FILE_BIN);
         if(filehandle!=INVALID_HANDLE)
           {
            //--- Sonuç[] dizisinin içeriğini bir dosyaya kaydetme
            FileWriteArray(filehandle,result,0,ArraySize(result));
            //--- Dosyayı kapatma
            FileClose(filehandle);
           }
         else
            Print("Error in FileOpen. Error code =",GetLastError());
        }
      else
         PrintFormat("Downloading '%s' failed, error code %d",url,res);
     }
  }
```

SendFTP

"FTP" sekmesinin ayarlar penceresinde belirtilen adrese dosya yollar.

```
bool  SendFTP(
   string  filename,           // ftp ile gönderilecek dosya
   string  ftp_path=NULL       // ftp kataloğu
   );
```

Parametreler

filename

[in] Gönderilen dosyanın ismi.

ftp_path=NULL

[in] FTP kataloğu. Eğer bir konum belirtilmemişse, ayarlar penceresinde tarif edilen konum kullanılır.

Dönüş değeri

Başarısızlık durumunda 'false'.

Not

Gönderilen dosya *terminal_dizini\MQL5\files* konumunda veya bunun alt klasörlerinde yer almalıdır. Eğer FTP adresi ve/veya şifresi ayarlarda belirtilmemişse, gönderim gerçekleşmez.

SendFTP() function does not work in the Strategy Tester.

SendMail

"Email" sekmesinin ayarlar penceresinde belirtilen adrese bir email gönderir.

```
bool  SendMail(
   string  subject,        // başlık
   string  some_text       // email metni
   );
```

Parametreler

subject

[in] Email başlığı.

some_text

[in] Email gövdesi.

Dönüş değeri

true - bir email, göndeme kuruğuna eklenmişse; aksi durumda - false.

Not

Ayarlar kısmında gönderme yasaklanmış olabilir, email adresi atlanmış da olabilir. Hata bilgisi için GetLastError() fonksiyonunu çağırın.

SendMail() function does not work in the Strategy Tester.

SendNotification

Bildirimler sekmesinde MetaQuotes ID'si belirtilen mobil terminallerine anlık bildirimler yolar.

```
bool  SendNotification(
   string  text          // Bildirim metni
   );
```

Parametreler

text

[in] Bildirimin metni. Mesaj uzunluğu 255 karakteri geçmemeli.

Dönüş değeri

Bildirim terminalden başarıyla gönderilmişse 'true'; başarısızlık durumunda ise 'false' değerine dönüş yapar. Başarısız olan bir anlık bildirim sonrasında kontrol yapılırken, GetLastError () fonksiyonu şu hatalardan birini dönebilir:

- 4515 - ERR_NOTIFICATION_SEND_FAILED,
- 4516 - ERR_NOTIFICATION_WRONG_PARAMETER,
- 4517 - ERR_NOTIFICATION_WRONG_SETTINGS,
- 4518 - ERR_NOTIFICATION_TOO_FREQUENT.

Not

SendNotification() fonksiyonu için katı kullanım kısıtlamaları getirilmiştir: saniyede iki çağrıdan fazlası ve dakikada 10 çağrıdan fazlası gerçekleştirilemez. Kulanım frekansının görüntülenmesi dinamiktir. Kısıtlamaların çiğnenmesi durumunda fonksiyon devre dışı bırakılacaktır.

SendNotification() function does not work in the Strategy Tester.

Müşteri terminalinin global değişkenleri

Global değişkenlerle çalışmak için bir grup fonksiyon bulunmaktadır.

Müşteri terminalinin global değişkenleri, MQL5 programının global düzeyinde bildirilen değişkenlerle karıştırılmamalıdır.

Global değişkenler, son erişimlerinin ardından 4 hafta boyunca müşteri terminali tarafından saklanır, sonra otomatik olarak silinirler. Global değişkene yapılan erişim, sadece yeni bir değerin ayarlanması anlamına gelmez, aynı zamanda global değişkenin değerinin okunması anlamına da gelir.

Müşteri terminalinin global değişkenleri, terminalde çalıştırılmış tüm programlarca erişilebilir durumdadır.

Fonksiyon	Eylem
GlobalVariableCheck	Belirtilen isimdeki bir global değişkenin varlığını kontrol eder
GlobalVariableTime	Global değişkene yapılan son erişimin zamanına dönüş yapar
GlobalVariableDel	Bir global değişkeni siler
GlobalVariableGet	Bir global değişkenin değerine dönüş yapar
GlobalVariableName	Global değişkenler listesindeki sıra sayısına göre global bir değişken ismine dönüş yapar
GlobalVariableSet	Bir global değişkene, yeni bir değer ayarlar
GlobalVariablesFlush	Tüm global değişkenlerin değerlerini zorla diske yazar
GlobalVariableTemp	Bir global değişkene, terminalin sadece mevcut oturumunda geçerli olacak yeni bir değer ayarlar
GlobalVariableSetOnCondition	Bir global değişkene, koşula bağlı olarak yeni bir değer ayarlar
GlobalVariablesDeleteAll	Belirtilen ön eke sahip global değişkeni siler
GlobalVariablesTotal	Global değişkenlerin toplam sayısına dönüş yapar

GlobalVariableCheck

Belirtilen isimdeki bir global değişkenin varlığını kontrol eder

```
bool  GlobalVariableCheck(
   string  name      // Global değişkenin ismi
   );
```

Parametreler

name

[in] Global değişkenin ismi.

Dönüş değeri

Söz konusu global değişken mevcutsa "true", deilse "false" değerine dönüş yapar.

Global değişkenler, son erişimlerinin ardından 4 hafta boyunca müşteri terminali tarafından saklanır, sonra otomatik olarak silinirler.

Ayrıca Bakınız

GlobalVariableTime()

GlobalVariableTime

Global değişkene yapılmış son erişimin zamanının değerine dönüş yapar.

```
datetime  GlobalVariableTime(
   string  name      // isim
   );
```

Parametreler

name

[in] Global değişkenin ismi.

Dönüş değeri

Fonksiyon, belirtilen global değişkene yapılmış son erişimin zamanına dönüş yapar. Değer alma amacıyla bir değişkenin çağrılması aynı zamanda bir erişim sayılır. Hata ile ilgili bilgi almak için, GetLastError() fonksiyonunu çağırın.

Not

Global değişkenler, son erişimlerinin ardından 4 hafta boyunca müşteri terminali tarafından saklanır. Daha sonra otomatik olarak silinirler.

Ayrıca Bakınız

GlobalVariableCheck()

GlobalVariableDel

Müşteri terminalinden bir global değişkeni siler.

```
bool  GlobalVariableDel(
   string  name      // Global değişkenin ismi
   );
```

Parametreler

name

[in] Global değişkenin ismi.

Dönüş değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapacaktır. Hata ile ilgili bilgi almak için, GetLastError() fonksiyonunun çağrılması gerekir.

Not

Global değişkenler, son erişimlerinin ardından 4 hafta boyunca müşteri terminali tarafından saklanır, sonra otomatik olarak silinirler.

GlobalVariableGet

Müşteri terminalinde mevcut bulunan bir global değişkenin değerine dönüş yapar. Fonksiyonun 2 çeşidi bulunmaktadır.

1. Global değişkenin değerine hemen dönüş yapar.

```
double  GlobalVariableGet(
   string  name      // Global değişkenin ismi
   );
```

2. Fonksiyonun başarı durumuna göre, 'true' veya 'false' değerine dönüş yapar. Başarı durumunda müşteri terminalinin global değişkeni, ikinci parametreye referansla geçirilen bir değişkene yerleştirilir

```
bool  GlobalVariableGet(
   string  name,              // Global değişkenin ismi
   double& double_var         // Global değişkenin değerini içerecek olan değişken
   );
```

Parametreler

name

[in] Global değişkenin ismi.

double_var

[out] Global değişkende saklanan değeri alacak olan double tipi hedef değişkeni.

Dönüş değeri

Mevcut global değişkenin değeri veya hata durumunda 0 değeri. Hata ile ilgili daha detaylı bilgi almak için GetLastError() fonksiyonunu çağırın.

Not

Global değişkenler, son erişimlerinin ardından 4 hafta boyunca müşteri terminali tarafından saklanır, sonra otomatik olarak silinirler.

GlobalVariableName

Sıra numarasına göre, global değişkenin ismine dönüş yapar.

```
string  GlobalVariableName(
   int  index      // Global değişkenin, global değişkenler listesindeki numarası
   );
```

Parametreler

index

[in] Global değişkenler listesindeki sıra numarası. Sıfıra eşit veya sıfırdan büyük ve GlobalVariablesTotal() değerinden küçük olmalı.

Dönüş değeri

Global değişkenler listesindeki sıra sayısına göre global bir değişken ismi. Hata ile ilgili daha detaylı bilgi almak için GetLastError() fonksiyonunu çağırın.

Not

Global değişkenler, son erişimlerinin ardından 4 hafta boyunca müşteri terminali tarafından saklanır, sonra otomatik olarak silinirler.

GlobalVariableSet

Bir global değişken için yeni bir değer ayarlar. Değişken mevcut değilse, sistem tarafından yeni bir global değişken oluşturulur.

```
datetime  GlobalVariableSet(
   string  name,       // Global değişken ismi
   double  value       // Ayarlanacak değer
   );
```

Parametreler

name

[in] Global değişkenin ismi.

value

[in] Yeni sayısal değer.

Dönüş değeri

Başarılı olması durumunda son değiştirilme zamanına, aksi durumda 0 değerine dönüş yapar. Hata hakkında daha detaylı bilgi almak için GetLastError() fonksiyonunu çağırın.

Not

A global variable name should not exceed 63 characters. Global değişkenler, son erişimlerinin ardından 4 hafta boyunca müşteri terminali tarafından saklanır, sonra otomatik olarak silinirler.

GlobalVariablesFlush

Tüm global değişkenlerin içeriklerini zorla diske kaydeder.

```
void  GlobalVariablesFlush();
```

Dönüş değeri

Dönüş değeri yok.

Not

İşlem bittiğinde, terminal tüm global değişkenleri yazar ama veri ani bir bilgisayar işlem hatası ile kaybolabilir. Bu fonksiyon, beklenmedik bir durumda global değişkenlerin kaydedilme sürecinin kontrolünü bağımsız olarak sağlar.

GlobalVariableTemp

Bu fonksiyon, geçici bir global değişken oluşturmayı dener. Eğer değişken mevcut değilse, sistem tarafından yeni bir geçici global değişken oluşturulur.

```
bool  GlobalVariableTemp(
   string  name      // Global değişkenin ismi
   );
```

Parametreler

name

[in] Geçici global değişkenin ismi.

Dönüş değeri

Başarı durumunda fonksiyon 'true' değerine, aksi durumda 'false' değerine dönüş yapacaktır. Hata ile ilgili bilgi almak için, GetLastError() fonksiyonunu çağırın.

Not

Geçici global değişkenler, sadece müşteri terminali çalışırken mevcutturlar; terminal kapatıldığında otomatik olarak silinirler. GlobalVariablesFlush() fonksiyonunun çalıştırılması sırasında geçici global değişkenlerin diske yazılmayacağını not edin.

Bir geçici global değişken oluşturulduktan sonra, tıpkı normal bir müşteri terminali global değişkeni gibi erişilebilir ve değiştirilebilir.

GlobalVariableSetOnCondition

Global değişkenin geçerli değeri, üçüncü parametre olarak belirtilen check_value değerine eşitse, değişken için yeni değer ayarlar. Eğer hiç global değişken yoksa, fonksiyon bir hata oluşturur ERR_GLOBALVARIABLE_NOT_FOUND (4501) ve 'false' dönüşü yapar.

```
bool  GlobalVariableSetOnCondition(
   string  name,            // Global değişkenin ismi
   double  value,           // Koşul "true" ise, değişkenin yeni değeri
   double  check_value      // Kontrol değeri koşulu
   );
```

Parametreler

name

[in] Global değişkenin ismi.

value

[in] Yeni değer.

check_value

[in] Global değişkenin mevcut değerini kontrol etmek için değer.

Dönüş değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapacaktır. Hata hakkında daha detaylı bilgi almak için GetLastError() fonksiyonunu çağırın. Eğer mevcut değer, check_value değerinden küçükse, fonksiyon 'false' dönüşü yapar.

Not

Fonksiyon bir global değişkene atomik erişim sağlar, bu yüzden müşteri terminali içinde aynı anda birden çok Uzman Danışman arasında mutex düzenlemek için kullanılabilir.

GlobalVariablesDeleteAll

Müşteri terminalinin global değişkenlerini siler.

```
int  GlobalVariablesDeleteAll(
   string     prefix_name=NULL,     // Ön ek ile başlayan tüm global değişken isimleri
   datetime   limit_data=0          // Bu tarihten önce değiştirilmiş tüm global değiş
   );
```

Parametreler

prefix_name=NULL

[in] Silinmesi istenen ön ek ile başlayan tüm global değişken isimleri. NULL veya boş değer şeklinde belirtildiğinde, kritere uyan tüm değişkenler silinir.

limit_data=0

[in] Son değiştirilme zamanına göre, global değişkenin seçimi için tarih. Fonksiyon, bu tarihten önce değiştirilmiş global değişkenleri siler. Eğer parametre sıfıra eşitse, ilk kriteri sağlayan tüm değişkenler silinir.

Dönüş değeri

Silinen değişkenlerin sayısı.

Not

Eğer iki seçenek de sıfıra eşitse (prefix_name = NULL ve limit_data = 0), fonksiyon, terminaldeki tüm global değişkenleri siler. İki parametre de belirtilmişse, parametrelere karşılık gelen global değişkenler silinir.

Global değişkenler, son erişimlerinin ardından 4 hafta boyunca müşteri terminali tarafından saklanır, sonra otomatik olarak silinirler.

GlobalVariablesTotal

Müşteri terminalinin global değişkenlerinin toplam sayısını verir.

```
int  GlobalVariablesTotal();
```

Dönüş değeri

Global değişkenlerin sayısı.

Not

Global değişkenler, son erişimlerinin ardından 4 hafta boyunca müşteri terminali tarafından saklanır, sonra otomatik olarak silinirler. Global değişkene yapılan çağrı, sadece yeni bir değerin ayarlanması anlamına gelmez, aynı zamanda global değişkenin değerinin okunması anlamına da gelir.

Dosya Fonksiyonları

Dosyalarla çalışmak için tasarlanmış fonksiyonlar grubu.

Dosya işlemleri, MQL5 dili içerisinde güvenlik amacıyla sıkı şekilde kontrol edilmektedir. İşlemleri, MQL5 araçları ile yürütülen dosyalar, dosya güvenlik-ortamı (sandbox) dışında yer alamaz.

Çalışan dosyaların yerleştirilebileceği iki konum (alt-konum) bulunmaktadır:

- terminal_veri_klasörü\MQL5\FILES\ (görüntülemek için terminal menüsünden "Dosya" - "Veri dosyasını aç" seçimlerini yapın);
- Bilgisayara kurulmuş tüm terminaller için ortak klasör - genellikle şu konumda yer alır C:\Documents and Settings\All Users\Application Data\MetaQuotes\Terminal\Common\Files.

Bu katalog (dosya yolu) isimleri, TerminalInfoString() fonksiyonunu ENUM_TERMINAL_INFO_STRING sayımıyla kullanarak elde edilebilir:

```
//--- Terminal verisini depolayan klasör
   string terminal_data_path=TerminalInfoString(TERMINAL_DATA_PATH);
//--- Tüm terminaller için ortak klasör
   string common_data_path=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
```

Diğer konumlardaki dosyalarla çalışmaya izin verilmez.

Dosya fonksiyonları "adlandırılmış kanallarla" çalışabilmenizi sağlar. Bunun için, FileOpen() fonksiyonunu uygun bir parametre ile çağırmanız gerekir.

Fonksiyon	Eylem
FileSelectDialog	Bir dosya veya klasör açma/oluşturma iletişim kutusu oluşturun
FileFindFirst	Belirtilen filtreye göre belli bir konumda dosya araması başlatır
FileFindNext	FileFindFirst() fonksiyonu ile yapılan aramayı devam ettirir
FileFindClose	Arama işleyicisini kapatır
FileOpen	Belirli bir isme ve bayrağa sahip olan dosyayı açar
FileDelete	Belirtilen bir dosyayı siler
FileFlush	Giriş/çıkış tamponunda kalan tüm veriyi bir diske yazar
FileGetInteger	Dosyaya dair bir tamsayı özelliğini alır
FileIsEnding	Okuma sürecinde bir dosyanın sonunu tanımlar
FileIsLineEnding	Okuma sürecinde bir metin dosyasındaki bir satırın sonunu tanımlar
FileClose	Açılmış bir dosyayı kapatır
FileIsExist	Bir dosyanın varlığını kontrol eder
FileCopy	Orjinal dosyayı, yerel veya paylaşımlı bir klasörden, başka bir dosyaya kopyalar
FileMove	Bir dosyayı taşır veya yeniden adlandırır

Fonksiyon	Eylem
FileReadArray	BIN tipi bir dosyadan, string harici herhangi tipteki dizileri okur
FileReadBool	CSV dosyasındaki bir dizgiyi ayırıcı işarete kadar (veya satır sonuna kadar) okur ardından okunan dizgiyi bool tipine dönüştürür
FileReadDatetime	CSV dosyasındaki, "YYYY.MM.DD HH:MM:SS", "YYYY.MM.DD" veya "HH:MM:SS" biçimlerinden birindeki dizgiyi okur ve bir datetime değerine dönüştürür
FileReadDouble	Dosya işaretçisinin mevcut konumundan double tipi değeri okur
FileReadFloat	Dosya işaretçisinin mevcut konumundan float tipi bir değeri okur
FileReadInteger	Dosya işaretçisinin mevcut konumundan short veya char tipi bir değeri okur
FileReadLong	Dosya işaretçisinin mevcut konumundan long tipi bir değeri okur
FileReadNumber	CSV dosyasındaki bir dizgiyi, mevcut konumdan ayırıcı işarete kadar (veya satır sonuna kadar) okur, ardından bu dizgiyi double tipine dönüştürür
FileReadString	Bir dosyadaki dizgiyi, dosya işaretçisinin mevcut konumundan başlayarak okur
FileReadStruct	Bir yapıya parametre olarak geçirilmiş ikili dosyanın içeriğini okur
FileSeek	Dosya işaretçisinin konumunu, belirtilen bayt sayısı ile, belirlenen konuma taşır
FileSize	Söz konusu açık dosyanın boyutuna dönüş yapar
FileTell	Söz konusu açık dosyanın, mevcut dosya işaretçisi konumuna dönüş yapar
FileWrite	CSV veya TXT tipi dosyaya veri yazar
FileWriteArray	BIN tipi bir dosyaya herhangi bir tipte (string hariç) diziler yazar
FileWriteDouble	İkili bir dosya içine, işaretçinin mevcut konumundan başlayarak double tipi bir değer yazar
FileWriteFloat	İkili bir dosya içine, işaretçinin mevcut konumundan başlayarak float tipi bir değer yazar file
FileWriteInteger	İkili bir dosya içine, işaretçinin mevcut konumundan başlayarak int tipi bir değer yazar file
FileWriteLong	İkili bir dosya içine, işaretçinin mevcut konumundan başlayarak long tipi bir değer yazar
FileWriteString	Bir BIN veya TXT dosyasının içine, işaretçinin mevcut konumundan başlayarak string tipi bir değer yazar
FileWriteStruct	İkili bir dosya içine, işaretçinin mevcut konumundan başlayarak bir yapının içeriklerini yazar

Fonksiyon	Eylem
FileLoad	Belirtilen ikili dosyadaki tüm verileri parametre şeklinde geçirilen sayısal diziye yazar
FileSave	Parametre şeklinde geçirilen dizinin tüm elemanlarını bir ikili dosyaya yazar
FolderCreate	Files dizini içinde (common_flag değerine bağlı olarak) bir klasör oluşturur
FolderDelete	Seçilen klasörü siler. Klasör boş değilse silinemez
FolderClean	Belirtilen klasördeki tüm dosyaları siler

Dosya, yazma amaçlı olarak FileOpen() fonksiyonu ile açılmışsa, dizinde bulunmayan alt-klasörler, oluşturulacaktır.

FileSelectDialog

Bir dosya veya klasör açma/oluşturma iletişim kutusu oluşturun.

```
int  FileSelectDialog(
   string   caption,              // pencere başlığı
   string   initial_dir,          // başlangıç dizini
   string   filter,               // uzantı filtresi
   uint     flags,                // bayrak kombinasyonu
   string&  filenames[],          // dosya adlarını içeren dizi
   string   default_filename      // varsayılan dosya adı
   );
```

Parametreler

caption

[in] İletişim kutusu penceresi başlığı.

initial_dir

[in] İçeriği iletişim kutusunda görüntülenecek olan MQL5\Files dizinindeki başlangıç dizini adı. Değer NULL ise, iletişim kutusunda MQL5\Files görüntülenir.

filter

[in] Seçim iletişim penceresinde görüntülenecek dosyaların uzantı filtresi. Diğer formatlardaki dosyalar gizlenir.

flags

[in] İletişim penceresi modunu tanımlayan bayrakların kombinasyonu. Bayraklar aşağıdaki gibi tanımlanır:
FSD_WRITE_FILE - dosya açma iletişim kutusu;
FSD_SELECT_FOLDER - yalnızca klasör seçimine izin ver;
FSD_ALLOW_MULTISELECT - birden fazla dosya seçimine izin ver;
FSD_FILE_MUST_EXIST - seçilen dosyalar mevcut olmalıdır;
FSD_COMMON_FOLDER - dosya tüm müşteri terminallerinin \Terminal\Common\Files ortak klasöründe bulunur.

filenames[]

[out] Seçili dosyaların/klasörlerin isimlerinin yerleştirileceği dizgeler dizisi.

default_filename

[in] Varsayılan dosya/klasör adı. Belirtilirse, açık iletişim kutusuna otomatik olarak bir ad eklenir ve sınama sırasında *filenames[]* dizisine geri döndürülür.

Geri dönüş değeri

Başarılı yürütme durumunda, fonksiyon, adları *filenames[]*'den elde edilebilen seçili dosyaların sayısını geri döndürür. Kullanıcı iletişim kutusunu bir dosya seçmeden kapatırsa fonksiyon 0 değerini döndürür. Başarısız yürütme durumunda, 0'dan küçük bir değer döndürülür. Hata kodu GetLastError() kullanılarak elde edilebilir.

Not

Güvenlik nedeniyle, dosyalarla çalışmak MQL5 dilinde katı bir biçimde denetlenir. MQL5 dili aracılığıyla dosya işlemlerinde kullanılan dosyalar, dosya sandboxının dışında (yani, MQL5\Files dizininin dışında) bulunamaz.

initial_dir adı, müşteri terminalinin dizinindeki MQL5\Files'da (veya sınama durumunda test_agent_directory\MQL5\Files'da) aranır. Bayraklar arasından FSD_COMMON_FOLDER ayarlanırsa, başlangıç dizininin aranması tüm müşteri terminalleri \Terminal\Common\Files ortak klasöründe gerçekleştirilir.

filter parametresi geçerli dosyaları belirtir ve "<açıklama 1>|<uzantı 1>|<açıklama 2>|<uzantı 2>..." biçiminde ayarlanmalıdır, örneğin, "Metin dosyaları (*.txt)|*.txt|Tüm dosyalar (*.*)|*.*". İlk uzantı "Metin dosyaları (*.txt)|*.txt" varsayılan dosya tipi olarak seçilir.

filter=NULL ise, iletişim penceresindeki dosya seçimi maskesi "Tüm Dosyalar (*.*)|*.*|" olur.

default_filename parametresi ayarlanırsa, FileSelectDialog()'u çağırmak 1 değerini geri döndürürken, *default_filename* değerinin kendisi görselleştirilmemiş sınama sırasında *filenames[]* dizisine kopyalanır.

Özel göstergelerde fonksiyonun kullanılması önerilmez, çünkü FileSelectDialog() çağrısı, kullanıcının yanıtını beklerken göstergenin yürütme iş parçasını tüm zaman boyunca askıya alır. Her bir sembol için tüm göstergeler tek bir iş parçacığında yürütüldüğünden, bu sembol için tüm zaman dilimlerinin tüm grafikleri askıya alınır.

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- istemci terminallerinin ortak klasöründen indirilecek metin dosyalarının adları
   string filenames[];
   if(FileSelectDialog("İndirilecek dosyaları seçin", NULL,
                       "Text files (*.txt)|*.txt|All files (*.*)|*.*",
                       FSD_ALLOW_MULTISELECT|FSD_COMMON_FOLDER, filenames, "data.txt")
     {
      //--- seçilen her dosyanın adını göster
      int total=ArraySize(filenames);
      for(int i=0; i<total; i++)
         Print(i, ": ", filenames[i]);
     }
   else
     {
     Print("Files not selected");
     }
//---
  }
```

Ayrıca bakınız

FileOpen, FileIsExist, FileDelete, FileMove, FolderCreate, FolderDelete, FolderClean, Flags of opening files

FileFindFirst

Bu fonksiyon, belirtilen konumdaki dosyalar veya alt-dizinler için belirtilen filtreye göre arama başlatır.

```
long  FileFindFirst(
   const string   file_filter,          // Dizgi - arama filtresi
   string&        returned_filename,    // Bulunacak dosyanın veya alt-dizinin ismi
   int            common_flag=0         // Aramayı biçimini tanımlar
   );
```

Parametreler

file_filter

[in] Arama filtresi. Herhangi bir alt-dizin (veya birden çok bitişik alt-dizin), aramanın yapılacağı \Files dizinine bağlı olarak, filtre içinde belirtilebilir.

returned_filename

[out] Dönüş parametresi - başarılı sonuç halinde, bulunan ilk dosyanın veya alt-dizinin ismi. 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] Dosyanın konumunu belirleyen bayrak. common_flag = FILE_COMMON ise dosya, tüm kullanıcılar için paylaşılan bir klasöre \Terminal\Common\Files, aksi durumda bir yerel klasöre yerleştirilmiştir.

Dönüş değeri

Başarılı sonuç durumunda, FileFindNext() fonksiyonu ile yapılacak daha sonraki aramalarda kullanılmak için, dosyanın veya alt-dizinin tanımlayıcı değerine dönüş yapar. Aksi durumda, filtreye karşılık gelen bir dosya veya <t3>alt-dizin</t3> bulunamamışsa (örneğin - dizin boş ise), INVALID_HANDLE değerine dönüş yapar. Arama tamamlandıktan sonra, işleyici FileFindClose() fonksiyonu ile kapatılmalıdır.

Not

Dosya işlemleri, MQL5 dili içerisinde güvenlik amacıyla sıkı şekilde kontrol edilmektedir. İşlemleri, MQL5 araçları ile yürütülen dosyalar, dosya güvenlik-ortamı (sandbox) dışında yer alamaz.

Örnek:

```
//--- 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!");
  }
```

Ayrıca Bakınız

FileFindNext, FileFindClose

FileFindNext

Bu fonksiyon, FileFindFirst() ile başlatılmış bir aramayı devam ettirir.

```
bool  FileFindNext(
   long      search_handle,          // Arama işleyicisi
   string&   returned_filename       // Aranan dosyanın veya alt-dizinin ismi
   );
```

Parametreler

search_handle

[in] FileFindFirst() tarafından alınan arama işleyicisi.

returned_filename

[out] Bulunan bir sonraki dosyanın veya alt-dizinin ismi. 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.

Dönüş değeri

Başarı durumunda 'true', aksi durumda 'false' dönüşü yapar.

Örnek:

```
//--- script çalıştırıldığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- filtre
input string InpFilter="*";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   int    i=1;
//--- yerel klasörün kök dizininde arama işleyicisini al
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- FileFindFirst() fonksiyonu başarılı şekilde uygulanmış mı kontrol et
   if(search_handle!=INVALID_HANDLE)
     {
      //--- döngüye geçirilen isimler dosya ismi mi, yoksa konum (dizin) ismi mi konti
      do
        {
         ResetLastError();
         //--- eğer bu bir dosya ise, fonksiyon 'true' dönüşü yapacak, eğer bir konum
         FileIsExist(file_name);
         PrintFormat("%d : %s isim = %s",i,GetLastError()==ERR_FILE_IS_DIRECTORY ? "Ko
         i++;
        }
      while(FileFindNext(search_handle,file_name));
```

```
      //--- arama işleyicisini seç
      FileFindClose(search_handle);
     }
   else
      Print("Dosyalar bulunamadı!");
  }
```

Ayrıca Bakınız

FileFindFirst, FileFindClose

FileFindClose

Arama işleyicisini kapatır.

```
void  FileFindClose(
   long  search_handle      //  Arama işleyicisini
   );
```

Parametreler

search_handle

[in] FileFindFirst() tarafından alınan arama işleyicisi.

Dönüş değeri

Dönüş değeri yok.

Not

Bu fonksiyon, sistem kaynaklarını boşaltmak için çağrılmalıdır.

Örnek:

```
//--- script çalıştırıldığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- filtre
input string InpFilter="*";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string file_name;
   int    i=1;
//--- yerel klasörün kök dizininde arama işleyicisini al
   long search_handle=FileFindFirst(InpFilter,file_name);
//--- FileFindFirst() fonksiyonu başarılı şekilde uygulanmış mı kontrol et
   if(search_handle!=INVALID_HANDLE)
     {
      //--- döngüye geçirilen isimler dosya ismi mi, yoksa konum (dizin) ismi mi konti
      do
        {
         ResetLastError();
         //--- eğer bu bir dosya ise, fonksiyon 'true' dönüşü yapacak, eğer bir konum
         FileIsExist(file_name);
         PrintFormat("%d : %s isim = %s",i,GetLastError()==5018 ? "Konum" : "Dosya",fi
         i++;
        }
      while(FileFindNext(search_handle,file_name));
      //--- arama işleyicisini seç
      FileFindClose(search_handle);
     }
```

```
   else
      Print("Dosyalar bulunamadı!");
  }
```

Ayrıca Bakınız

FileFindFirst, FileFindNext

FileIsExist

Bir dosyanın var olup olmadığını kontrol eder.

```
bool  FileIsExist(
   const string  file_name,        // Dosya ismi
   int           common_flag=0     // Arama alanı
   );
```

Parametreler

file_name

[in] Kontrolü yapılacak dosyanın ismi

common_flag=0

[in] Dosyanın konumunu belirleyen bayrak. common_flag = FILE_COMMON ise dosya, tüm kullanıcılar için paylaşılan bir klasöre \Terminal\Common\Files, aksi durumda bir yerel klasöre yerleştirilmiştir.

Dönüş değeri

Dosya mevcut ise 'true' değerine dönüş yapar.

Not

Kontrol edilen dosyanın bir alt-dizin olması mümkündür. Bu durumda FileIsExist() fonksiyonu false dönüşü yapacaktır ve 5018 kodlu hata, _LastError değişkenine kaydedilecektir - "Bu bir dizin, dosya değil" (FileFindFirst fonksiyonunun kullanımına bakınız).

Dosya işlemleri, MQL5 dili içerisinde güvenlik amacıyla sıkı şekilde kontrol edilmektedir. İşlemleri, MQL5 araçları ile yürütülen dosyalar, dosya güvenlik-ortamı (sandbox) dışında yer alamaz.

common_flag = FILE_COMMON ise, fonksiyon, dosyayı tüm terminallerin ortak klasöründe arayacaktır \Terminal\Common\Files, aksi durumda ise yerel bir klasörde arama yapacaktır (MQL5\Files veya sınama durumunda MQL5\Tester\Files).

Örnek:

```
//--- script çalıştığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- eski dosya için tarih
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;       // isimleri depolamak için bir değişken
   string   filter="*.txt"; // dosyaları aramak için filtre
   datetime create_date;     // dosya oluşturulma zamanı
   string   files[];         // dosya isimlerinin listesi
   int      def_size=25;     // varsayılan dizi büyüklüğü
   int      size=0;          // dosyaların sayısı
```

```
//--- dizi için bellek tahsis et
   ArrayResize(files,def_size);
//--- yerel klasörün kök dizinindeki arama işleyicisini al
   long search_handle=FileFindFirst(filter,file_name);
//--- FileFindFirst() uygulaması başarılı mı kontrol et
   if(search_handle!=INVALID_HANDLE)
     {
      //--- dosyaları döngüde ara
      do
        {
         files[size]=file_name;
         //--- dizi büyüklüğünü artır
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- hata değerini sıfırla
         ResetLastError();
         //--- dosya oluşturulma zamanını al
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- dosya eski mi kontrol et
         if(create_date<InpFilesDate)
           {
            PrintFormat("%s dosyası silinmiş!",file_name);
            //--- eski dosyayı sil
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- arama işleyicisini kapa
      FileFindClose(search_handle);
     }
   else
     {
      Print("Dosyalar bulunamadı!");
      return;
     }
//--- kalan dosyaları kontrol et
   PrintFormat("Sonuçlar:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("%s dosyası mevcut!",files[i]);
      else
         PrintFormat("%s dosyası silinmiş!",files[i]);
     }
  }
```

Ayrıca Bakınız

FileFindFirst

FileOpen

Bu fonksiyon, belirtilen isme ve bayrağa sahip olan dosyayı açar.

```
int  FileOpen(
   string  file_name,           // Dosya ismi
   int     open_flags,          // Bayrak kombinasyonu
   short   delimiter='\t',      // Ayırıcı
   uint    codepage=CP_ACP      // Kod sayfası
   );
```

Parametreler

file_name

[in] Dosya ismi alt klasörleri içerebilir. Eğer dosya yazma amacıyla açılmışsa, bu alt klasörler mevcut değillerse oluşturulurlar.

open_flags

[in] bayrak kombinasyonu, dosya için kullanılacak işlem modunu belirtir. Bayraklar aşağıdaki gibi tanımlanır:
FILE_READ dosya okuma amacıyla açılır
FILE_WRITE dosya yazma amacıyla açılır
FILE_BIN ikili okuma-yazma modu (dizgi dönüşümü yoktur)
FILE_CSV csv tipi dosya (kaydedilen şeyler, unicode veya ansi tipi dizgilere dönüştürülür, ve bir ayırıcı ile ayrılır)
FILE_TXT basit metin dosyasıdır (csv tipine benzer ama ayırıcı hesaba katılmaz)
FILE_ANSI ANSI tipi satırlar (tek baytlık semboller halinde)
FILE_UNICODE UNICODE tipi satırlar (iki baytlık karakterler)
FILE_SHARE_READ birkaç programdan paylaşımlı okuma
FILE_SHARE_WRITE birkaç program ile paylaşımlı yazma
FILE_COMMON tüm terminallerin ortak klasöründeki dosyanın konumu \Terminal\Common\Files

delimiter='\t'

[in] txt veya csv dosyasında ayırıcı alarak kullanılacak değer. csv dosyasının ayırıcısı belirtilmemişse, varsayılan olarak sekme kullanılır. txt dosyasının ayırıcısı belirtilmemişse, ayırıcı kullanılmaz. Ayırıcı açıkça 0 olarak ayarlanmışsa, yine ayırıcı kullanılmaz.

codepage=CP_ACP

[in] Kod sayfasının değeri. Uygun sabitleri sağlayan, en çok kullanılan kod sayfaları için.

Dönüş değeri

Eğer dosya başarılı şekilde açılmışsa, fonksiyon tanıtıcı değere dönüş yapar, bu da daha sonra dosya verilerine erişmek için kullanılır. Başarısızlık durumunda INVALID_HANDLE değerine dönüş yapar.

Not

Dosya işlemleri, MQL5 dili içerisinde güvenlik amacıyla sıkı şekilde kontrol edilmektedir. İşlemleri, MQL5 araçları ile yürütülen dosyalar, dosya güvenlik-ortamı (sandbox) dışında yer alamaz.

Dosya belirli bir kodlamayla okunacaksa FILE_ANSI bayrağını ayarladığınızdan emin olun (kod sayfası parametresi ilebir kod sayfası değeri belirtilir). Belirtilen FILE_ANSI bayrağı yoksa, metin dosyası hiç dönüşüm olmadan Unicode'da okunur.

Dosya, müşteri terminali klasörünün MQL5\files alt-klasöründe açılır (veya sınama durumunda sınama_birimi_dizini\MQL5\files klasöründe). FILE_COMMON, bayraklar arasında belirtilmişse, dosya tüm MetaTrader 5 müşteri terminallerinin ortak klasöründe açılır.

"Adlandırılmış kanallar" şu kurallara göre açılabilir:

- Kanal (pipe) ismi bir dizgidir ve şu şekle sahip olmalıdır "\\servername\pipe\pipename", burada "servername" ağdaki sunucunun ismi, "pipename" ise kanalın ismidir. Kanallar aynı bilgisayar üzerinde kullanılıyorsa, sunucu ismi atlanabilir ve bunun yerine bir nokta kullanılır: "\\.\pipe\pipename". Kanala bağlanmaya çalışan müşteri, kanalın ismini bilmelidir.
- FileFlush() ve FileSeek() fonksiyonları, kanaldan veya kanala yapılacak ardışık okuma veya yazma işlemlerinin arasında çağrılmalıdır.

Gösterilen dizgilerde '\' özel sembolü kullanılır. Bu yüzden bir MQL5 uygulaması içinde bir isim yazarken, '\' sembolü çift olarak kullanılmalıdır. Yani yukarıdaki örnek, kod içerisinde şu görünüme sahip olmalıdır: "\\\\servername\\pipe\\pipename".

Adlandırılmış kanallarla çalışma hakkında daha fazla bilgi, "DLL'ler kullanmadan Named Pipes kullanarak Metatrader 5 ile iletişim kurma" isimli makalede bulunabilir.

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- hatalı dosya açma yöntemi
   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("Salt adres kullanılarak dosya aзma işlemi başarısız oldu ");
      Print("Hata kodu ",GetLastError());
     }

//--- "dosya güvenlik ortamında" çalışmanın doğru şekli
   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("FileOpen işlemi başarısız oldu, hata ",GetLastError());
//--- MQL5\Files\ dizinine iliştirilmiş bir klasör oluşturulması için başka bir örnek
   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("Dosya şu klasörde oluşturulmalı "+terminal_data_path+"\\"+subfolder);
     }
   else Print("Dosya açma başarısız, hata ",GetLastError());
  }
```

Ayrıca Bakınız

Bir Kod Sayfasının Kullanımı, FileFindFirst, FolderCreate, Dosya açma bayrakları

FileClose

FileOpen() fonksiyonu ile açılmış olan dosyayı kapatır.

```
void  FileClose(
   int  file_handle      // Dosya tanıtıcı değeri
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

Dönüş değeri

Dönüş değeri yok.

Örnek:

```
//--- script çalıştığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- giriş parametreleri
input string InpFileName="file.txt";     // dosya ismi
input string InpDirectoryName="Data";    // dizin ismi
input int    InpEncodingType=FILE_ANSI; // ANSI=32 veya UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- kullanacağımız dosya adresini çıktıla
   PrintFormat("%s\\Files\\ klasöründe çalışılıyor",TerminalInfoString(TERMINAL_DATA_
//--- hata değerini sıfırla
   ResetLastError();
//--- dosyayı okuma amaçlı olarak aç (dosya mevcut değilse hata oluşacak)
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEn
   if(file_handle!=INVALID_HANDLE)
     {
      //--- dosya içeriğini çıktıla
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- dosyayı kapat
      FileClose(file_handle);
     }
   else
      PrintFormat("Hata, kod = %d",GetLastError());
  }
```

FileCopy

Yerel veya paylaşılan bir klasördeki orjinal dosyayı başka bir dosyaya kopyalar.

```
bool  FileCopy(
   const string  src_file_name,      // Kaynak dosyasının ismi
   int           common_flag,        // Konum
   const string  dst_file_name,      // Hedef dosyanın ismi
   int           mode_flags          // Erişim modu
   );
```

Parametreler

src_file_name

[in] Kopyalanacak dosyanın ismi.

common_flag

[in] Dosyanın konumunu belirleyen bayrak. common_flag = FILE_COMMON ise dosya, tüm kullanıcılar için paylaşılan bir klasöre \Terminal\Common\Files, aksi durumda ise yerel bir klasöre yerleştirilmiştir (örneğin, *common_flag=0*).

dst_file_name

[in] Sonuç dosyasının ismi.

mode_flags

[in] Erişim bayrakları. Bu parametre sadece iki bayrak içerebilir: FILE_REWRITE ve/veya FILE_COMMON - diğer bayraklar gözardı edilir. Eğer dosya zaten mevcutsa ve FILE_REWRITE bayrağı belirtilmemişse, dosya yeniden yazılmaz ve false dönüşü yapar.

Dönüş değeri

Başarısız sonuç durumunda false değerine dönüş yapar.

Not

Dosya işlemleri, MQL5 dili içerisinde güvenlik amacıyla sıkı şekilde kontrol edilmektedir. İşlemleri, MQL5 araçları ile yürütülen dosyalar, dosya güvenlik-ortamı (sandbox) dışında yer alamaz.

Dosya zaten mevcutsa, kopyalama işlemi FILE_REWRITE bayrağının mode_flags parametresindeki varlığına bağlı olacaktır.

Örnek:

```
//--- script çalıştırıldığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- giriş parametreleri
input string InpSrc="source.txt";       // kaynak
input string InpDst="destination.txt";  // kopya
input int    InpEncodingType=FILE_ANSI; // ANSI=32 veya UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
```

```
  {
//--- kaynak içeriğini görüntüle (mevcut olmalı)
   if(!FileDisplay(InpSrc))
      return;
//--- hedef dosyası zaten mevcut mu kontrol et (oluşturulamayabilir)
   if(!FileDisplay(InpDst))
     {
      //--- hedef dosya mevcut değil, FILE_REWRITE bayrağı olmadan kopyala (doğru kopy
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("Dosya kopyalandı!");
      else
         Print("Dosya kopyalanamadı!");
     }
   else
     {
      //--- Hdef dosya zaten mevcut, FILE_REWRITE bayrağı olmadan kopyalamayı dene (ha
      if(FileCopy(InpSrc,0,InpDst,0))
         Print("Dosya kopyalandı!");
      else
         Print("Dosya kopyalanamadı!");
      //--- InpDst dosyasının içeriği aynı kaldı
      FileDisplay(InpDst);
      //--- bir kez daha kopyala, bu sefer FILE_REWRITE bayrağını kullan (dosya mevcut
      if(FileCopy(InpSrc,0,InpDst,FILE_REWRITE))
         Print("Dosya kopyalandı!");
      else
         Print("Dosya kopyalanamadı!");
     }
//--- InpSrc dosya kopyasını al
   FileDisplay(InpDst);
  }
//+------------------------------------------------------------------+
//| Dosya içeriklerini oku                                           |
//+------------------------------------------------------------------+
bool FileDisplay(const string file_name)
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- dosyayı aç
   int file_handle=FileOpen(file_name,FILE_READ|FILE_TXT|InpEncodingType);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- dosya içeriğini döngü içinde görüntüle
      Print("+----------------------+");
      PrintFormat("Dosya ismi = %s",file_name);
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      Print("+----------------------+");
      //--- dosyayı kapat
```

```
      FileClose(file_handle);
      return(true);
     }
//--- dosya açılamadı
   PrintFormat("%s açılamadı, hata = %d",file_name,GetLastError());
   return(false);
  }
```

FileDelete

Müşteri terminalinin yerel klasöründe belirtilen dosyayı siler.

```
bool  FileDelete(
   const string  file_name,      // Silinecek dosya adı
   int           common_flag=0   // Silinecek dosyanın konumu
   );
```

Parametreler

file_name

[in] Dosya ismi.

common_flag=0

[in] Dosya konumunu belirleyen bayrak. common_flag = FILE_COMMON ise dosya, tüm kullanıcılar için paylaşılan bir klasöre \Terminal\Common\Files, aksi durumda bir yerel klasöre yerleştirilmiştir.

Dönüş değeri

Başarısız sonuç durumunda false değerine dönüş yapar.

Not

Dosya işlemleri, MQL5 dili içerisinde güvenlik amacıyla sıkı şekilde kontrol edilmektedir. İşlemleri, MQL5 araçları ile yürütülen dosyalar, dosya güvenlik-ortamı (sandbox) dışında yer alamaz.

Müşteri terminalinin yerel klasöründe belirtilen bir dosyayı siler (MQL5\Files veya sınama durumunda MQL5\Tester\Files). common_flag = FILE_COMMON ise, fonksiyon dosyayı paylaşılan tüm terminal klasörlerinden siler.

Örnek:

```
//--- script çalıştığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- eski dosya için tarih
input datetime InpFilesDate=D'2013.01.01 00:00';
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // isimleri depolamak için bir değişken
   string   filter="*.txt"; // dosyaları aramak için filtre
   datetime create_date;    // dosya oluşturulma zamanı
   string   files[];        // dosya isimlerinin listesi
   int      def_size=25;    // varsayılan dizi büyüklüğü
   int      size=0;         // dosyaların sayısı
//--- dizi için bellek tahsis et
   ArrayResize(files,def_size);
//--- yerel klasörün kök dizinindeki arama işleyicisini al
   long search_handle=FileFindFirst(filter,file_name);
//--- FileFindFirst() uygulaması başarılı mı kontrol et
```

```
   if(search_handle!=INVALID_HANDLE)
     {
      //--- dosyaları döngüde ara
      do
        {
         files[size]=file_name;
         //--- dizi büyüklüğünü artır
         size++;
         if(size==def_size)
           {
            def_size+=25;
            ArrayResize(files,def_size);
           }
         //--- hata değerini sıfırla
         ResetLastError();
         //--- dosya oluşturulma zamanını al
         create_date=(datetime)FileGetInteger(file_name,FILE_CREATE_DATE,false);
         //--- dosya eski mi kontrol et
         if(create_date<InpFilesDate)
           {
            PrintFormat("%s dosyası silinmiş!",file_name);
            //--- eski dosyayı sil
            FileDelete(file_name);
           }
        }
      while(FileFindNext(search_handle,file_name));
      //--- arama işleyicisini kapa
      FileFindClose(search_handle);
     }
   else
     {
      Print("Dosyalar bulunamadı!");
      return;
     }
//--- kalan dosyaları kontrol et
   PrintFormat("Sonuçlar:");
   for(int i=0;i<size;i++)
     {
      if(FileIsExist(files[i]))
         PrintFormat("%s dosyası mevcut!",files[i]);
      else
         PrintFormat("%s dosyası silinmiş!",files[i]);
     }
  }
```

FileMove

Bir dosyayı yerel veya paylaşılan bir klasörden, başka bir klasöre taşır.

```
bool  FileMove(
   const string  src_file_name,     // Taşınacak dosyanın ismi
   int           common_flag,       // Konum
   const string  dst_file_name,     // Hedef dosyasının ismi
   int           mode_flags         // Erişim modu
   );
```

Parametreler

src_file_name

[in] Taşınacak/yeniden isimlendirilecek dosyanın ismi.

common_flag

[in] Dosyanın konumunu belirleyen bayrak. common_flag = FILE_COMMON ise dosya, tüm kullanıcılar için paylaşılan bir klasöre \Terminal\Common\Files, Aksi durumda bir yerel klasöre yerleştirilmiş (*common_flag=0*).

dst_file_name

[in] İşlem sonundaki dosya ismi

mode_flags

[in] Erişim bayrakları. Bu parametre sadece iki bayrak içerebilir: FILE_REWRITE ve/veya FILE_COMMON - diğer bayraklar gözardı edilir. Eğer dosya zaten mevcutsa ve FILE_REWRITE bayrağı belirtilmemişse, dosya yeniden yazılmaz ve false dönüşü yapar.

Dönüş değeri

Başarısız sonuç durumunda false değerine dönüş yapar.

Not

Dosya işlemleri, MQL5 dili içerisinde güvenlik amacıyla sıkı şekilde kontrol edilmektedir. İşlemleri, MQL5 araçları ile yürütülen dosyalar, dosya güvenlik-ortamı (sandbox) dışında yer alamaz.

Dosya zaten mevcutsa, kopyalama işlemi FILE_REWRITE bayrağının mode_flags parametresindeki varlığına bağlı olacaktır.

Örnek:

```
//--- script çalıştırıldığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- giriş parametreleri
input string InpSrcName="data.txt";
input string InpDstName="newdata.txt";
input string InpSrcDirectory="SomeFolder";
input string InpDstDirectory="OtherFolder";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

```
void OnStart()
  {
   string local=TerminalInfoString(TERMINAL_DATA_PATH);
   string common=TerminalInfoString(TERMINAL_COMMONDATA_PATH);
//--- dosya adreslerini al
   string src_path;
   string dst_path;
   StringConcatenate(src_path,InpSrcDirectory,"//",InpSrcName);
   StringConcatenate(dst_path,InpDstDirectory,"//",InpDstName);
//--- kaynak dosyası mevcut mu kontrol et (mevcut değilse - çık)
   if(FileIsExist(src_path))
      PrintFormat("%s dosyası %s\\Files\\%s konumunda mevcut",InpSrcName,local,InpSrcI
   else
     {
      PrintFormat("Hata, %s kaynak dosyası bulunamadı",InpSrcName);
      return;
     }
//--- Sonuç dosyası zaten mevcut mu kontrol et
   if(FileIsExist(dst_path,FILE_COMMON))
     {
      PrintFormat("%s dosyası %s\\Files\\%s klasöründe mevcut",InpDstName,common,InpDs
      //--- dosya mevcut, taşıma işlemi FILE_REWRITE bayrağı ile gerçekleştirilmeli
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON|FILE_REWRITE))
         PrintFormat("%s dosyası taşındı",InpSrcName);
      else
         PrintFormat("Hata! Kod = %d",GetLastError());
     }
   else
     {
      PrintFormat("%s dosyası, %s\\Files\\%s klasöründe mevcut değil",InpDstName,commo
      //--- dosya mevcut değil, taşıma işlemi FILE_REWRITE bayrağı olmadan gerçekleşti
      ResetLastError();
      if(FileMove(src_path,0,dst_path,FILE_COMMON))
         PrintFormat("%s dosyası taşındı",InpSrcName);
      else
         PrintFormat("Hata! Kod = %d",GetLastError());
     }
//--- dosya taşındı, şimdi kontrol edelim
   if(FileIsExist(dst_path,FILE_COMMON) && !FileIsExist(src_path,0))
      Print("Başarılı!");
   else
      Print("Hata!
  }
```

Ayrıca Bakınız

FileIsExist

FileFlush

Giriş/çıkış tamponunda kalan tüm veriyi diske yazar.

```
void  FileFlush(
   int  file_handle      // Dosya tanıtıcı değeri
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

Dönüş değeri

Dönüş değeri yok.

Not

Bir dosya yazarken, veriler dosyada belli bir zaman sonra oluşabilir. Veriyi dosyaya anında kaydetmek amacıyla, FileFlush() fonksiyonunu kullanın. Bu fonksiyonun kullanılmaması durumunda, verinin bir kısmı hala diske yazılmamış olabilir, dosya FileClose() fonksiyonu ile kapatıldığında zorla yazılacaktır.

Fonksiyon, yazılan verinin kesin ve doğru bir değere sahip olması durumunda kullanılmalıdır. Sık yapılan fonksiyon çağrısının, programın işlem zamanını etkileyeceği not edilmelidir.

FileFlush() fonksiyonu, bir dosya üzerinde yapılan okuma ve yazma işlemlerinin arasında çağrılmalıdır.

Örnek:

```
//--- script çalıştığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- Yazılacak dosyanın ismi
input string InpFileName="example.csv"; // dosya ismi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- hata değerini sıfırla
   ResetLastError();
//--- dosyayı aç
   int file_handle=FileOpen(InpFileName,FILE_READ|FILE_WRITE|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- veriyi dosyaya yaz
      for(int i=0;i<1000;i++)
        {
         //--- Yazma fonksiyonunu çağır
         FileWrite(file_handle,TimeCurrent(),SymbolInfoDouble(Symbol(),SYMBOL_BID),Sy
         //--- her 128-inci tekrarda veriyi kaydet
```

```
         if((i & 127)==127)
           {
            //--- artık veri dosyaya yazılı durumda ve herhangi bir kritik hata durumu
            FileFlush(file_handle);
            PrintFormat("i = %d, OK",i);
           }
         //--- 0.01 saniye durakla
         Sleep(10);
        }
      //--- dosyayı kapat
      FileClose(file_handle);
     }
   else
      PrintFormat("Hata, kod = %d",GetLastError());
  }
```

Ayrıca Bakınız

FileClose

FileGetInteger

Dosyanın bir tam-sayı özelliğini alır. Fonksiyonun iki çeşidi bulunmaktadır.

1. Özelliği dosya tanıtıcısı ile al.

```
long  FileGetInteger(
   int                          file_handle,    // Dosya tanıtıcısı
   ENUM_FILE_PROPERTY_INTEGER   property_id     // Özellik tanımlayıcı
   );
```

2. Özeliği dosya ismi ile al.

```
long  FileGetInteger(
   const string                 file_name,            // Dosya ismi
   ENUM_FILE_PROPERTY_INTEGER   property_id,          // Özellik yaanımlayıcı
   bool                         common_folder=false   // Dosya bir yerel klasörde (fals
   );                                                 // veya tüm terminallerin ortak
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

file_name

[in] Dosya ismi.

property_id

[in] Dosya özelliğinin tanımlayıcısı. Bu değer ENUM_FILE_PROPERTY_INTEGER sayımının değerlerinden biri olabilir. Fonksiyonun ikinci versiyonunun kullanılması durumunda, sadece şu özellikleri alabilirsiniz: FILE_EXISTS, FILE_CREATE_DATE, FILE_MODIFY_DATE, FILE_ACCESS_DATE ve FILE_SIZE.

common_folder=false

[in] Dosya konumunu gösterir. Parametre false ise, terminal veri klasörü gösterilir Aksi durumda dosyanın, tüm terminallerin ortak klasöründe \Terminal\Common\Files (FILE_COMMON) olduğu varsayılır.

Dönüş değeri

Özellik değeri. Hata durumunda -1 dönüşü yapar. Hata kodunu almak için GetLastError() fonksiyonunu kullanın.

Özelliklerin alınması için dosya ismi belirtilmişse, fonksiyon 5018 numaralı hatayı alır (ERR_MQL_FILE_IS_DIRECTORY) bu durumda dönüş değeri doğru olacaktır.

Not

Fonksiyon, hata kodunu sürekli değiştirir. Başarılı sonuç durumunda, hata kodu NULL değerini alır.

Örnek:

```
//--- script çalıştırıldığında giriş parametrelerinin penceresini göster
#property script_show_inputs
```

```
//--- giriş parametreleri
input string InpFileName="data.csv";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string path=InpDirectoryName+"//"+InpFileName;
   long   l=0;
//--- dosyayı aç
   ResetLastError();
   int handle=FileOpen(path,FILE_READ|FILE_CSV);
   if(handle!=INVALID_HANDLE)
     {
      //--- Dosya hakkındaki tüm bilgiyi çıktıla
      Print(InpFileName," dosya bilgisi:");
      FileInfo(handle,FILE_EXISTS,l,"bool");
      FileInfo(handle,FILE_CREATE_DATE,l,"tarih");
      FileInfo(handle,FILE_MODIFY_DATE,l,"tarih");
      FileInfo(handle,FILE_ACCESS_DATE,l,"tarih");
      FileInfo(handle,FILE_SIZE,l,"diğer");
      FileInfo(handle,FILE_POSITION,l,"diğer");
      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");
      //--- dosyayı kapat
      FileClose(handle);
     }
   else
      PrintFormat("%s dosya açılmadı, Hata Kodu = %d",InpFileName,GetLastError());
  }
//+------------------------------------------------------------------+
//| Dosya özelliğinin değerini görüntüle                             |
//+------------------------------------------------------------------+
void FileInfo(const int handle,const ENUM_FILE_PROPERTY_INTEGER id,
              long l,const string type)
  {
//--- özellik değerini al
   ResetLastError();
   if((l=FileGetInteger(handle,id))!=-1)
     {
      //--- değer alındı, şimdi doğru şekilde görüntüle
      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("Hata, Kod = ",GetLastError());
  }
```

Ayrıca Bakınız

Dosya İşlemleri, Dosya Özellikleri

FileIsEnding

Okuma süreci içinde bir dosyanın sonunu tanımlar.

```
bool  FileIsEnding(
   int  file_handle      // Dosya tanıtıcı değeri
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

Dönüş değeri

Okuma süreci içerisinde veya dosya işaretçisini taşırken, dosyanın sonuna ulaşılmışsa 'true' değerine dönüş yapar.

Not

Dosyanın sonunu tanımlamak amacıyla, fonksiyon dosyadaki bir sonraki dizgiyi okumayı dener. Eğer böyle bir dizgi bulunmuyorsa, fonksiyon 'true' dönüşü yapar, aksi durumda ise 'false' dönüşü yapar.

Örnek:

```
//--- script çalıştığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- giriş parametreleri
input string InpFileName="file.txt";     // dosya ismi
input string InpDirectoryName="Data";    // dizin ismi
input int    InpEncodingType=FILE_ANSI; // ANSI=32 veya UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- kullanacağımız dosya adresini çıktıla
   PrintFormat("%s\\Files\\ klasöründe çalışılıyor",TerminalInfoString(TERMINAL_DATA_P
//--- hata değerini sıfırla
   ResetLastError();
//--- dosyayı okuma amaçlı olarak aç (dosya mevcut değilse hata oluşacak)
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEn
   if(file_handle!=INVALID_HANDLE)
     {
      //--- dosya içeriğini çıktıla
      while(!FileIsEnding(file_handle))
         Print(FileReadString(file_handle));
      //--- dosyayı kapat
      FileClose(file_handle);
     }
   else
      PrintFormat("Hata, kod = %d",GetLastError());
```

```
}
```

FileIsLineEnding

Bu fonksiyon, okuma sürecinde bir metin dosyasındaki satırın sonunu tanımlar.

```
bool  FileIsLineEnding(
   int  file_handle      // Dosya tanıtıcı değeri
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

Dönüş değeri

txt veya csv dosyasının okunması sürecinde satır sonuna gelinmişse (CR-LF karakterleri), true dönüşü yapar.

Örnek (Burada, FileWriteString fonksiyonunun uygulama örneğinden elde edilen dosya kullanılmıştır)

```
//+------------------------------------------------------------------+
//|                                        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
//--- verinin okunması için gereken parametreler
input string InpFileName="RSI.csv";   // dosya ismi
input string InpDirectoryName="Data"; // dizin ismi
//--- gösterge tamponları
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- aşırı-alım değişkenleri
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
//--- aşırı-satım değişkenleri
```

```
int      ovs_ind=0;
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ön tanımlı olarak dizi büyüklüğü değişkenleri
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- diziler için bellek tahsis et
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- dosyayı aç
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası, okuma için müsait",InpFileName);
      PrintFormat("Dosya yolu: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      double value;
      //--- dosyadan veri oku
      while(!FileIsEnding(file_handle))
        {
         //--- dizgideki ilk değeri oku
         value=FileReadNumber(file_handle);
         //--- fonksiyon sonucuna göre farklı dizilere oku
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- dosyayı kapat
      FileClose(file_handle);
      PrintFormat("Veri yazıldı, %s dosyası kapatıldı",InpFileName);
     }
   else
     {
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- dizilerin bağlanması
   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);
//---- çizelgede görüntülenmesi istenmeyen gösterge değerlerini ayarla
```

```
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Dosyadaki dizgi verisini oku                                     |
//+------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- dizginin veya sayfanın sonuna kadar oku
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- sayıyı okuduktan sonra kaydır
      if(flag)
         FileReadNumber(file_handle);
      //--- mevcut tarihi kaydet
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- gerekirse dizi büyüklüğünü artır
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- ilk tekrarı geç
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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);
//--- hala işlenmemiş çubuklar için döngü
```

```
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- varsayılan olarak 0
      open_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      close_buff[i]=0;
      color_buff[i]=0;
      //--- tarihlerden biri hala mevcut mu kontrol et
      if(ovb_ind<ovb_size)
         for(int j=ovb_ind;j<ovb_size;j++)
           {
            //--- tarihler aynıysa çubuk aşırı-alım bölgesinde
            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 - kırmızı renk
               color_buff[i]=0;
               //--- sayacı artır
               ovb_ind=j+1;
               break;
              }
           }
      //--- hala her hangi bir veri var mı kontrol et
      if(ovs_ind<ovs_size)
         for(int j=ovs_ind;j<ovs_size;j++)
           {
            //--- tarihler kesişiyorsa, çubuk aşırı-satım bölgesinde
            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 - mavi renk
               color_buff[i]=1;
               //--- sayacı artır
               ovs_ind=j+1;
               break;
              }
           }
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

```
//| ChartEvent event handler                                         |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
                  )
  {
//--- gösterge çizgi genişliğini ölçeğe göre değiştir
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

Ayrıca Bakınız

FileWriteString

FileReadArray

Bu fonksiyon, herhangi tipteki dizilerden (dizgi ve dinamik dizi içermeyen bir yapı dizisi olabilir) oluşan BIN tipli bir dosyadan okuma yapar.

```
uint  FileReadArray(
   int     file_handle,              // Dosya tanıtıcısı
   void&   array[],                  // Kayıt yapılacak dizi
   int     start=0,                  // yazım yapmak için dizideki başlangıç konumu
   int     count=WHOLE_ARRAY         // okunacak eleman sayısı
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

array[]

[out] Verinin yükleneceği dizi.

start=0

[in] Dizinin yazılması için başlangıç konumu.

count=WHOLE_ARRAY

[in] Okunacak elemanların sayısı. Varsayılan olarak tüm diziyi okur (count=WHOLE_ARRAY).

Dönüş değeri

Okunan elemanların sayısı.

Not

Dizgilerden oluşan diziler sadece TXT tipi dosyalardan okunabilirler. Fonksiyon, gerektiği takdirde dizinin büyüklüğünü artırmayı dener.

Örnek (burada kullanılan dosya, FileWriteArray fonksiyonunun kullanımından elde edilmiştir)

```
//--- script çalıştırıldığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- giriş parametreleri
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| Fiyat verisini depolamak için bir yapı                           |
//+------------------------------------------------------------------+
struct prices
  {
   datetime          date; // tarih
   double            bid;  // satış fiyatı
   double            ask;  // alış fiyatı
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- yapı dizisi
   prices arr[];
//--- dosya yolu
   string path=InpDirectoryName+"//"+InpFileName;
//--- dosyayı aç
   ResetLastError();
   int file_handle=FileOpen(path,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      //--- dosyadan tüm verileri diziye oku
      FileReadArray(file_handle,arr);
      //--- dizi büyüklüğü değerini al
      int size=ArraySize(arr);
      //--- diziden verileri yazdır (çıktıla)
      for(int i=0;i<size;i++)
         Print("Tarih = ",arr[i].date," Satış = ",arr[i].bid," Alış = ",arr[i].ask);
      Print("Toplam veri = ",size);
      //--- dosyayı kapat
      FileClose(file_handle);
     }
   else
      Print("Dosya açma başarısız, hata ",GetLastError());
  }
```

Ayrıca Bakınız

Değişkenler, FileWriteArray

FileReadBool

Bu fonksiyon, CSV tipi bir dosyada, ayırıcı konumuna kadar (veya metin dosyasının sonuna kadar) olan bir dizgiyi okur ve sonra okunan dizgiyi bool tipi bir değere dönüştürür.

```
bool  FileReadBool(
   int  file_handle    // Dosya tanıtıcısı
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

Dönüş değeri

Okunan satır "true" veya "false" değerlerine ayarlanabileceği gibi, "0" veya "1" içeren sembolik bir ifadeyle de ayarlanabilir. Sıfır olmayan her değer mantıksal "true" değerine dönüştürülür. Fonksiyon, dönüştürülmüş değere dönüş yapar.

Örnek (burada kullanılan dosya, FileWrite fonksiyonu örneğinden elde edilmiştir)

```
//+------------------------------------------------------------------+
//|                                            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
//--- verinin okunması için gereken parametreler
input string InpFileName="MACD.csv";  // dosya ismi
input string InpDirectoryName="Data"; // dizin ismi
//--- global değişkenler
int      ind=0;       // indis
double   upbuff[];    // yukarı yönlü okların gösterge tamponu
```

```
double   downbuff[];  // aşağı yönlü okların gösterge tamponu
bool     sign_buff[]; // sinyal dizisi (true - al, false - sat)
datetime time_buff[]; // sinyal geliş zamanının dizisi
int      size=0;      // sinyal dizilerinin boyutu
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dosyayı aç
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası okuma için açıldı",InpFileName);
      //--- ilk olarak, sinyallerin sayısını oku
      size=(int)FileReadNumber(file_handle);
      //--- diziler için bellek tahsis et
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- dosyadan veriyi oku
      for(int i=0;i<size;i++)
        {
         //--- sinyal zamanı
         time_buff[i]=FileReadDatetime(file_handle);
         //--- sinyal değeri
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- dosyayı kapat
      FileClose(file_handle);
     }
   else
     {
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- dizilerin bağlanması
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- PLOT_ARROW içinde kullanılacak sembol kodunu ayarla
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- çizelgede görüntülenmesi istenmeyen gösterge değerlerini ayarla
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

```
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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);
//--- hala işlenmemiş çubuklar için döngü
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- varsayılan olarak 0
      upbuff[i]=0;
      downbuff[i]=0;
      //--- hala, herhangi bir veri mevcut mu kontrol et
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- Eğer tarihler örtüşüyorsa, dosyadaki değeri kullan
            if(time[i]==time_buff[j])
              {
               //--- sinyale göre ok çiz
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- sayacı artır
               ind=j+1;
               break;
              }
           }
        }
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Ayrıca Bakınız

bool Tipi, FileWrite

FileReadDatetime

Bu fonksiyon, bir CSV dosyasından, "YYYY.MM.DD HH:MI:SS", "YYYY.MM.DD" veya "HH:MI:SS" biçimlerinden birine sahip bir dizgiyi okur ve ardından datetime tipine çevirir.

```
datetime  FileReadDatetime(
   int  file_handle    // Dosya tanıtıcısı
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

Dönüş değeri

datetime tipi değer.

Örnek (burada kullanılan dosya, FileWrite fonksiyonu örneğinden elde edilmiştir)

```
//+------------------------------------------------------------------+
//|                                        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
//--- verinin okunması için gereken parametreler
input string InpFileName="MACD.csv";  // dosya ismi
input string InpDirectoryName="Data"; // dizin ismi
//--- global değişkenler
int      ind=0;       // indis
double   upbuff[];    // yukarı yönlü okların gösterge tamponu
double   downbuff[];  // aşağı yönlü okların gösterge tamponu
bool     sign_buff[]; // sinyal dizisi (true - al, false - sat)
```

```
datetime time_buff[]; // sinyal geliş zamanının dizisi
int      size=0;      // sinyal dizilerinin boyutu
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dosyayı aç
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası okuma için açıldı",InpFileName);
      //--- ilk olarak, sinyallerin sayısını oku
      size=(int)FileReadNumber(file_handle);
      //--- diziler için bellek tahsis et
      ArrayResize(sign_buff,size);
      ArrayResize(time_buff,size);
      //--- dosyadan veriyi oku
      for(int i=0;i<size;i++)
        {
         //--- sinyal zamanı
         time_buff[i]=FileReadDatetime(file_handle);
         //--- sinyal değeri
         sign_buff[i]=FileReadBool(file_handle);
        }
      //--- dosyayı kapat
      FileClose(file_handle);
     }
   else
     {
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- dizilerin bağlanması
   SetIndexBuffer(0,upbuff,INDICATOR_DATA);
   SetIndexBuffer(1,downbuff,INDICATOR_DATA);
//--- PLOT_ARROW içinde kullanılacak sembol kodunu ayarla
   PlotIndexSetInteger(0,PLOT_ARROW,241);
   PlotIndexSetInteger(1,PLOT_ARROW,242);
//---- çizelgede görüntülenmesi istenmeyen gösterge değerlerini ayarla
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
   PlotIndexSetDouble(1,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
```

```
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);
//--- hala işlenmemiş çubuklar için döngü
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- varsayılan olarak 0
      upbuff[i]=0;
      downbuff[i]=0;
      //--- hala, herhangi bir veri mevcut mu kontrol et
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- Eğer tarihler örtüşüyorsa, dosyadaki değeri kullan
            if(time[i]==time_buff[j])
              {
               //--- sinyale göre ok çiz
               if(sign_buff[j])
                  upbuff[i]=high[i];
               else
                  downbuff[i]=low[i];
               //--- sayacı artır
               ind=j+1;
               break;
              }
           }
        }
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Ayrıca Bakınız

datetime Tipi, StringToTime, TimeToString, FileWrite

FileReadDouble

Bu fonksiyon, ikili bir dosyanın geçerli pozisyonundan bir kayan noktalı çifte-duyarlık sayısını (double) okur.

```
double  FileReadDouble(
   int  file_handle    // Dosya tanıtıcısı
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

Dönüş değeri

double tipli değer.

Not

Hata ile ilgili daha detaylı bilgi almak için, GetLastError() fonksiyonunu çağırın.

Örnek (burada kullanılan dosya, FileWriteDouble fonksiyonunun örneğinden elde edilmiştir)

```
//+------------------------------------------------------------------+
//|                                          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
//--- veri okuma parametreleri
input string InpFileName="MA.csv";     // dosya ismi
input string InpDirectoryName="Data"; // dizin ismi
//--- global değişkenler
int      ind=0;
int      size=0;
double   ma_buff[];
datetime time_buff[];
//--- gösterge tamponu
double   buff[];
```

```
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dosyayı aç
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası, okuma için müsait",InpFileName);
      PrintFormat("Dosya yolu: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- ilk önce, dosyada bulunan veri miktarını oku
      size=(int)FileReadDouble(file_handle);
      //--- diziler için bellek tahsis et
      ArrayResize(ma_buff,size);
      ArrayResize(time_buff,size);
      //--- dosyadan veriyi oku
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadDouble(file_handle);
         ma_buff[i]=FileReadDouble(file_handle);
        }
      //--- dosyayı kapat
      FileClose(file_handle);
      PrintFormat("Veri yazıldı, %s dosyası kapatıldı",InpFileName);
     }
   else
     {
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- diziyi, 0 indisli tampona bağla
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- çizelgede görüntülenmesi istenmeyen gösterge değerlerini ayarla
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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);
//--- hala işlenmemiş çubuklar için döngü
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- varsayılan olarak 0
      buff[i]=0;
      //--- hala her hangi bir veri var mı kontrol et
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- eğer tarihler örtüşüyorsa, dosyadaki değer kullanılır
            if(time[i]==time_buff[j])
              {
               buff[i]=ma_buff[j];
               //--- sayacı artır
               ind=j+1;
               break;
              }
           }
        }
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Ayrıca Bakınız

Reel tipler (double, float), StringToDouble, DoubleToString, FileWriteDouble

FileReadFloat

Bu fonksiyon, ikili dosya içindeki geçerli konumdan tek-duyarlıklı kayan noktalı sayı (float) okur.

```
float  FileReadFloat(
   int  file_handle    // Dosya tanıtıcısı
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

Dönüş değeri

float tipi değer.

Not

Hata ile ilgili daha detaylı bilgi almak için, GetLastError() fonksiyonunu çağırın.

Örnek (burada kullanılan dosya FileWriteFloat fonksiyonunun örneğinden elde edilmiştir)

```
//+------------------------------------------------------------------+
//|                                              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
//--- verinin okunması için gereken parametreler
input string InpFileName="Close.bin"; // dosya ismi
input string InpDirectoryName="Data"; // dizin ismi
//--- global değişkenler
int      ind=0;
int      size=0;
double   close_buff[];
datetime time_buff[];
//--- gösterge tamponları
double   buff[];
double   color_buff[];
//+------------------------------------------------------------------+
```

```
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- diziler için bellek tahsis et
   ArrayResize(close_buff,def_size);
   ArrayResize(time_buff,def_size);
//--- dosyayı aç
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası, okuma için müsait",InpFileName);
      PrintFormat("Dosya yolu: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- dosyadan veriyi oku
      while(!FileIsEnding(file_handle))
        {
         //--- zaman ve fiyat değerlerini oku
         time_buff[size]=(datetime)FileReadDouble(file_handle);
         close_buff[size]=(double)FileReadFloat(file_handle);
         size++;
         //--- taşan dizilerin büyüklüklerini artır
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(close_buff,def_size);
            ArrayResize(time_buff,def_size);
           }
        }
      //--- dosyayı kapat
      FileClose(file_handle);
      PrintFormat("Veriler okundu, %s dosyası kapatıldı",InpFileName);
     }
   else
     {
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- dizileri gösterge tamponlarına bağla
   SetIndexBuffer(0,buff,INDICATOR_DATA);
   SetIndexBuffer(1,color_buff,INDICATOR_COLOR_INDEX);
//---- çizelgede görüntülenmesi istenmeyen gösterge değerlerini ayarla
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
```

```
//+------------------------------------------------------------------+
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);
//--- hala işlenmemiş çubuklar için döngü
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- varsayılan olarak 0
      buff[i]=0;
      color_buff[i]=0; // varsayılan olarak kırmızı renk
      //--- hala, herhangi bir veri mevcut mu kontrol et
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- eğer tarihler örtüşüyorsa, dosyadaki değer kullanılır
            if(time[i]==time_buff[j])
              {
               //--- fiyatı al
               buff[i]=close_buff[j];
               //--- mevcut fiyat öncekini geçmişse, renk mavi olacak
               if(buff[i-1]>buff[i])
                  color_buff[i]=1;
               //--- sayacı artır
               ind=j+1;
               break;
              }
           }
        }
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Ayrıca Bakınız

Reel tipler (double, float), FileReadDouble, FileWriteFloat

FileReadInteger

Bu fonksiyon, bayt bazında belirtilen dosya işaretçisine bağlı olarak, dosya işaretçisinin mevcut konumundan int, short veya char tipli değer okur.

```
int  FileReadInteger(
   int  file_handle,         // Dosya tanıtıcı değeri
   int  size=INT_VALUE       // Bir tamsayının bayt olarak büyüklüğü
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

size=INT_VALUE

[in] Okunması gereken baytların sayısı (en fazla 4). Karşılık gelen sabitler verilmiştir: CHAR_VALUE = 1, SHORT_VALUE = 2 ve INT_VALUE = 4, böylece fonksiyon, char, short veya int tipli bir değerin tamamını okuyabilir.

Dönüş değeri

int tipli bir değer. Bu fonksiyonun sonucu açık bir şekilde hedef tipine, yani okunması gereken verinin tipine dönüştürülmelidir. int tipli bir değere dönüş yapılmışsa, kolaylıkla herhangi bir tamsayı tipine dönüştürülebilir. Dosya işaretçisi, okunan bayt sayısına göre kaydırma yapar.

Not

Okuma 4 bayttan az olduğunda, elde edilen sonuç her zaman pozitiftir. Bir veya iki baytlık bir okuma gerçekleştiriliyorsa, sayının işareti char (1 bayt) veya short (2 bayt) tiplerine yapılacak açık dönüşümle belirlenebilir. Karşılık gelen bir temel tip bulunmuyorsa, üç baytlık bir sayının işaretinin alınması önemsizdir.

Example (burada kullanılan dosya, FileWriteInteger fonksiyonunun örneğinden elde edilmiştir)

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

```
//--- verinin okunması için gereken parametreler
input string InpFileName="Trend.bin"; // dosya ismi
input string InpDirectoryName="Data"; // dizin ismi
//--- global değişkenler
int      ind=0;
int      size=0;
datetime time_buff[];
//--- gösterge tamponları
double   buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int def_size=100;
//--- dizi için bellek tahsis et
   ArrayResize(time_buff,def_size);
//--- dosyayı aç
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası, okuma için müsait",InpFileName);
      PrintFormat("Dosya yolu: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- ek değişkenler
      int    arr_size;
      uchar  arr[];
      //--- dosyadan veriyi oku
      while(!FileIsEnding(file_handle))
        {
         //--- zamanın yazılması için kaç adet sembol kullanıldığını öğren
         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);
         //--- zaman değerini kaydet
         time_buff[size]=StringToTime(CharArrayToString(arr));
         size++;
         //--- eğer taşma varsa dizilerin büyüklüklerini artır
         if(size==def_size)
           {
            def_size+=100;
            ArrayResize(time_buff,def_size);
           }
        }
      //--- dosyayı kapat
      FileClose(file_handle);
      PrintFormat("Veriler okundu, %s dosyası kapatıldı",InpFileName);
     }
```

```
   else
     {
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- diziyi gösterge tamponlarına bağla
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- çizelgede görüntülenmesi istenmeyen gösterge değerlerini ayarla
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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);
//--- hala işlenmemiş çubuklar için döngü
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- varsayılan olarak 0
      buff[i]=0;
      //--- hala, herhangi bir veri mevcut mu kontrol et
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- tarihler örtüşüyorsa dosyadaki değeri kullan
            if(time[i]==time_buff[j])
              {
               //--- fiyatı al
               buff[i]=close[i];
               //--- sayacı artır
               ind=j+1;
               break;
              }
           }
        }
```

```
      }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Ayrıca Bakınız

IntegerToString, StringToInteger, Tamsayı tipleri, FileWriteInteger

FileReadLong

Bu fonksiyon, ikili dosyanın geçerli konumundan long tipi (8 bayt) bir tamsayı değeri okur.

```
long  FileReadLong(
   int  file_handle    // Dosya tanıtıcısı
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

Dönüş değeri

long tipli değer.

Example (burada, FileWriteLong fonksiyonunun uygulama örneğinden elde edilen dosya kullanılmıştır)

```
//+------------------------------------------------------------------+
//|                                              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
//--- verinin okunması için gereken parametreler
input string InpFileName="Volume.bin"; // dosya ismi
input string InpDirectoryName="Data";  // dizin ismi
//--- global değişkenler
int      ind=0;
int      size=0;
long     volume_buff[];
datetime time_buff[];
//--- gösterge tamponları
double   buff[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
```

```
//--- dosyayı aç
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN);
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası, yazma amacıyla açıldı",InpFileName);
      PrintFormat("Dosya yolu: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- ilk önce, dosyada bulunan veri miktarını oku
      size=(int)FileReadLong(file_handle);
      //--- diziler için bellek tahsis et
      ArrayResize(volume_buff,size);
      ArrayResize(time_buff,size);
      //--- dosyadan veriyi oku
      for(int i=0;i<size;i++)
        {
         time_buff[i]=(datetime)FileReadLong(file_handle);
         volume_buff[i]=FileReadLong(file_handle);
        }
      //--- dosyayı kapat
      FileClose(file_handle);
      PrintFormat("Veriler okundu, %s dosyası kapatıldı",InpFileName);
     }
   else
     {
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- diziyi, 0 indisli gösterge tamponuna bağla
   SetIndexBuffer(0,buff,INDICATOR_DATA);
//---- çizelgede görüntülenecek gösterge değerlerini ayarla
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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);
```

```
//--- hala işlenmemiş çubuklar için döngü
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- varsayılan olarak 0
      buff[i]=0;
      //--- hala, herhangi bir veri mevcut mu kontrol et
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- tarihler örtüşüyorsa dosyadaki değeri kullan
            if(time[i]==time_buff[j])
              {
               buff[i]=(double)volume_buff[j];
               ind=j+1;
               break;
              }
           }
        }
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Ayrıca Bakınız

Tamsayı tipleri, FileReadInteger, FileWriteLong

FileReadNumber

Bu fonksiyon, CSV dosyasındaki bir dizgiyi ayırıcı işarete kadar (veya metin dizgisinin sonuna kadar) okur ardından okunan dizgiyi bool tipine dönüştürür.

```
double  FileReadNumber(
   int  file_handle    // Dosya tanıtıcısı
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

Dönüş değeri

double tipli değer.

Örnek (Burada, FileWriteString fonksiyonunun uygulama örneğinden elde edilen dosya kullanılmıştır)

```
//+------------------------------------------------------------------+
//|                                          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
//--- verinin okunması için gereken parametreler
input string InpFileName="RSI.csv";   // dosya ismi
input string InpDirectoryName="Data"; // dizin ismi
//--- gösterge tamponları
double   open_buff[];
double   high_buff[];
double   low_buff[];
double   close_buff[];
double   color_buff[];
//--- aşırı-alım değişkenleri
int      ovb_ind=0;
int      ovb_size=0;
datetime ovb_time[];
//--- aşırı-satım değişkenleri
```

```
int      ovs_ind=0;
int      ovs_size=0;
datetime ovs_time[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- ön tanımlı olarak dizi büyüklüğü değişkenleri
   int ovb_def_size=100;
   int ovs_def_size=100;
//--- diziler için bellek tahsis et
   ArrayResize(ovb_time,ovb_def_size);
   ArrayResize(ovs_time,ovs_def_size);
//--- dosyayı aç
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_CSV|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası, okuma için müsait",InpFileName);
      PrintFormat("Dosya yolu: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      double value;
      //--- dosyadan veri oku
      while(!FileIsEnding(file_handle))
        {
         //--- dizgideki ilk değeri oku
         value=FileReadNumber(file_handle);
         //--- fonksiyon sonucuna göre farklı dizilere oku
         if(value>=70)
            ReadData(file_handle,ovb_time,ovb_size,ovb_def_size);
         else
            ReadData(file_handle,ovs_time,ovs_size,ovs_def_size);
        }
      //--- dosyayı kapat
      FileClose(file_handle);
      PrintFormat("Veri yazıldı, %s dosyası kapatıldı",InpFileName);
     }
   else
     {
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- dizilerin bağlanması
   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);
//---- çizelgede görüntülenmesi istenmeyen gösterge değerlerini ayarla
```

```
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Dosyadaki dizgi verisini oku                                     |
//+------------------------------------------------------------------+
void ReadData(const int file_handle,datetime &arr[],int &size,int &def_size)
  {
   bool flag=false;
//--- dizginin veya sayfanın sonuna kadar oku
   while(!FileIsLineEnding(file_handle) && !FileIsEnding(file_handle))
     {
      //--- sayıyı okuduktan sonra kaydır
      if(flag)
         FileReadNumber(file_handle);
      //--- mevcut tarihi kaydet
      arr[size]=FileReadDatetime(file_handle);
      size++;
      //--- gerekirse dizi büyüklüğünü artır
      if(size==def_size)
        {
         def_size+=100;
         ArrayResize(arr,def_size);
        }
      //--- ilk tekrarı geç
      flag=true;
     }
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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);
//--- hala işlenmemiş çubuklar için döngü
```

```
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- varsayılan olarak 0
      open_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      close_buff[i]=0;
      color_buff[i]=0;
      //--- tarihlerden biri hala mevcut mu kontrol et
      if(ovb_ind<ovb_size)
         for(int j=ovb_ind;j<ovb_size;j++)
           {
            //--- tarihler aynıysa çubuk aşırı-alım bölgesinde
            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 - kırmızı renk
               color_buff[i]=0;
               //--- sayacı artır
               ovb_ind=j+1;
               break;
              }
           }
      //--- hala her hangi bir veri var mı kontrol et
      if(ovs_ind<ovs_size)
         for(int j=ovs_ind;j<ovs_size;j++)
           {
            //--- tarihler kesişiyorsa, çubuk aşırı-satım bölgesinde
            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 - mavi renk
               color_buff[i]=1;
               //--- sayacı artır
               ovs_ind=j+1;
               break;
              }
           }
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

```
//| ChartEvent event handler                                         |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam
                  )
  {
//--- gösterge çizgi genişliğini ölçeğe göre değiştir
   if(ChartGetInteger(0,CHART_SCALE)>3)
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,2);
   else
      PlotIndexSetInteger(0,PLOT_LINE_WIDTH,1);
  }
```

Ayrıca Bakınız

FileWriteString

FileReadString

Belirtilen dosyadaki dizgiyi dosya işaretçisinin mevcut konumundan başlayarak okur.

```
string  FileReadString(
   int  file_handle,     // Dosya tanıtıcısı
   int  length=-1        // Dizgi uzunluğu
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

length=-1

[in] Okunacak karakterlerin sayısı.

Dönüş değeri

Okunan satır (dizgi).

Not

Bir bin-dosyasından okuma yapılırken. okunacak dizginin uzunluğu belirtilmelidir. txt dosyasından okuma yapıldığında dizgi uzunluğunun belirtilmesi gerekmez ve dizgi, mevcut pozisyondan "\r\n" satır atlama karakterine kadar okunur. csv dosyasından okuma yapıldığında da, aynı şekilde, dizgi uzunluğunun belirtilmesi gerekmez ve dizgi mevcut pozisyondan ayırıcıya kadar veya metin dizgisinin sonuna kadar okunur

Eğer dosya FILE_ANSI bayrağı ile açılmışsa, o zaman satırlar Unicode'a dönüştürülür.

Example (burada kullanılan dosya, FileWriteInteger fonksiyonunun örneğinden elde edilmiştir)

```
//--- script çalıştırıldığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- verinin okunması için gereken parametreler
input string InpFileName="Trend.bin"; // dosya ismi
input string InpDirectoryName="Data"; // dizin ismi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- dosyayı aç
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası, okuma için müsait",InpFileName);
      PrintFormat("Dosya yolu: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- ek değişkenler
      int    str_size;
      string str;
```

```
      //--- dosyadan veriyi oku
      while(!FileIsEnding(file_handle))
        {
         //--- zamanın yazılması için kaç adet sembol kullanıldığını öğren
         str_size=FileReadInteger(file_handle,INT_VALUE);
         //--- dizgiyi oku
         str=FileReadString(file_handle,str_size);
         //--- dizgiyi çıktıla
         PrintFormat(str);
        }
      //--- dosyayı kapat
      FileClose(file_handle);
      PrintFormat("Veriler okundu, %s dosyası kapatıldı",InpFileName);
     }
   else
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
  }
```

Ayrıca Bakınız

string Tipi, Veri Dönüşümü, FileWriteInteger

FileReadStruct

İkili dosya içeriğini, mevcut konumdan başlayarak parametre olarak geçirilen bir yapıya okur.

```
uint  FileReadStruct(
   int           file_handle,        // dosya tanıtıcısı
   const void&   struct_object,      // içeriğin okunacağı hedef yapı
   int           size=-1             // bayt olarak yapı boyutu
   );
```

Parametreler

file_handle

[in] Açık bir bin dosyasının tanıtıcısı.

struct_object

[out] Söz konusu yapının nesnesi. Bu yapı, dizgiler, dinamik diziler veya sanal fonksiyonlar içermemelidir.

size=-1

[in] Okunacak bayt sayısı. Boyut belirtilmemişse veya belirtilen değer yapının boyutunu aşıyorsa, söz konusu yapının boyutu kullanılır.

Dönüş değeri

Başarılı sonuç durumunda, okunan bayt sayısına dönüş yapılır ve dosya işaretçisi, eşit sayıda bayt ile kaydırılır.

Örnek (Burada kullanılan dosya FileWriteStruct fonksiyonunun örneğinden elde edilmiştir)

```
//+------------------------------------------------------------------+
//|                                          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  "Candles"
#property indicator_type1   DRAW_CANDLES
#property indicator_color1  clrOrange
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
#property indicator_separate_window
//--- verileri almak için parametreler
input string  InpFileName="EURUSD.txt"; // dosya ismi
input string  InpDirectoryName="Data";  // dizin ismi
//+------------------------------------------------------------------+
```

```
//| Mum verilerini depolamak için bir yapı                           |
//+------------------------------------------------------------------+
struct candlesticks
  {
   double            open;  // açılış fiyatı
   double            close; // kapanış fiyatı
   double            high;  // yüksek fiyat
   double            low;   // düşük fiyat
   datetime          date;  // tarih
  };
//--- gösterge tamponları
double       open_buff[];
double       close_buff[];
double       high_buff[];
double       low_buff[];
//--- global değişkenler
candlesticks cand_buff[];
int          size=0;
int          ind=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
   int default_size=100;
   ArrayResize(cand_buff,default_size);
//--- dosyayı aç
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_BIN|FILE_
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası, okuma için müsait",InpFileName);
      PrintFormat("Dosya adresi: %s\\Files\\",TerminalInfoString(TERMINAL_COMMONDATA_P
      //--- dosyadan veriyi oku
      while(!FileIsEnding(file_handle))
        {
         //--- veriyi diziye yaz
         FileReadStruct(file_handle,cand_buff[size]);
         size++;
         //--- dizi taşmış mı kontrol et
         if(size==default_size)
           {
            //--- dizinin boyutlarını artır
            default_size+=100;
            ArrayResize(cand_buff,default_size);
           }
        }
      //--- dosyayı kapat
      FileClose(file_handle);
```

```
      PrintFormat("Veriler okundu, %s dosyası kapatıldı",InpFileName);
     }
   else
     {
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
      return(INIT_FAILED);
     }
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,open_buff,INDICATOR_DATA);
   SetIndexBuffer(1,high_buff,INDICATOR_DATA);
   SetIndexBuffer(2,low_buff,INDICATOR_DATA);
   SetIndexBuffer(3,close_buff,INDICATOR_DATA);
//--- boş değer
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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);
//--- hala işlenmemiş olan mumlar için bir döngü
   for(int i=prev_calculated;i<rates_total;i++)
     {
      //--- varsayılan olarak 0
      open_buff[i]=0;
      close_buff[i]=0;
      high_buff[i]=0;
      low_buff[i]=0;
      //--- hala, herhangi bir veri mevcut mu kontrol et
      if(ind<size)
        {
         for(int j=ind;j<size;j++)
           {
            //--- tarihler örtüşüyorsa dosyadaki değeri kullan
            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;
                //--- sayacı artır
                ind=j+1;
                break;
               }
            }
         }
      }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Ayrıca Bakınız

Yapılar ve sınıflar, FileWriteStruct

FileSeek

Dosya işaretçisinin konumunu, belirtilen konuma göre, belirtilen bayt sayısı kadar taşır.

```
bool  FileSeek(
   int                  file_handle,     // Dosya tanıtıcı değeri
   long                 offset,          // Bayt
   ENUM_FILE_POSITION   origin           // Referans konumu
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

offset

[in] Bayt bazında kaydırma değeri (negatif değer alabilir).

origin

[in] Kaydırma için başlangıç konumu. ENUM_FILE_POSITION sayımının değerlerinden biri olabilir.

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata ile ilgili bilgi almak için GetLastError() fonksiyonunu çağırın.

Not

FileSeek() fonksiyonunun çalışması sonucunda negatif kaydırma değeri elde ediliyorsa (dosyanın "sol sınırı" dışına çıkılırsa), dosya işaretçisi dosyanın başlangıcına ayarlanır.

Eğer konum, "sağ sınırın" ötesine ayarlanmışsa (dosya boyutundan büyükse), bir sonraki dosya yazımı dosyanın sonundan değil, belirtilen konumdan başlar Bu durumda, dosyanın önceki sonu ile ayarlanan pozisyonu arasına boş değerler yazılacaktır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                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"
//--- script çalıştırıldığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- giriş parametreleri
input string InpFileName="file.txt";     // dosya ismi
input string InpDirectoryName="Data";    // dizin ismi
input int    InpEncodingType=FILE_ANSI; // ANSI=32 veya UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

```
//+------------------------------------------------------------------+
void OnStart()
  {
//--- rassal sayıları oluşturmak için değişken değerini belirle
   _RandomSeed=GetTickCount();
//--- dizgilerin başlangıç noktası konumları için değişkenler
   ulong pos[];
   int   size;
//--- hata değerini sıfırla
   ResetLastError();
//--- dosyayı aç
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası, okuma için müsait",InpFileName);
      //--- dosyadaki her bir dizgi için başlangıç konumu al
      GetStringPositions(file_handle,pos);
      //--- dosyadaki dizgilerin sayısını tanımla
      size=ArraySize(pos);
      if(!size)
        {
         //--- dosyada dizgi yoksa dur
         PrintFormat("%s dosyası boş!",InpFileName);
         FileClose(file_handle);
         return;
        }
      //--- dizgi sayısını rassal olarak seç
      int ind=MathRand()%size;
      //--- konumu dizginin başlangıç noktasına kaydır
      if(FileSeek(file_handle,pos[ind],SEEK_SET)==true)
        {
      //--- ind sayısı ile dizgiyi oku ve çıktıla
         PrintFormat("%d numaralı dizgi metni: \"%s\"",ind,FileReadString(file_handle)
        }
      //--- dosyayı kapat
      FileClose(file_handle);
      PrintFormat("%s dosyası kapatıldı",InpFileName);
     }
   else
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
  }
//+------------------------------------------------------------------------------+
//| Bu fonksiyon, dosyadaki her dizgi için başlangıç konumları tanımlar          |
//| ve bunları arr dizisine yerleştirir                                          |
//+------------------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
  {
//--- varsayılan dizi büyüklüğü
   int def_size=127;
```

```
//--- dizi için bellek tahsis et
   ArrayResize(arr,def_size);
//--- dizgi sayacı
   int i=0;
//--- bu, dosyanın sonu değilse, hala en azından bir dizgi kalmıştır
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // dosya boş, çık
//--- kaydırmayı kodlama biçimine göre ve bayt bazında tanımla
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
      shift=1;
   else
      shift=2;
//--- dizgileri döngü içinde incele
   while(1)
     {
      //--- dizgiyi oku
      FileReadString(handle);
      //--- dosya sonunu kontrol et
      if(!FileIsEnding(handle))
        {
         //--- bir sonraki dizginin konumunu kaydet
         arr[i]=FileTell(handle)+shift;
         i++;
         //--- taşma varsa dizi büyüklüğünü artır
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // dosyanın sonuna gelindi, çık
     }
//--- dizinin gerçek büyüklüğünü tanımla
   ArrayResize(arr,i);
  }
```

FileSize

Dosya boyutu değerine bayt cinsinden dönüş yapar.

```
ulong  FileSize(
   int  file_handle    // Dosya tanıtıcısı
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

Dönüş değeri

int tipli değer.

Not

Hata ile ilgili bilgi almak için GetLastError() fonksiyonunu çağırın.

Örnek:

```
//--- script çalıştığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- giriş parametreleri
input ulong  InpThresholdSize=20;         // kilobayt bazında dosya boyutu
input string InpBigFolderName="big";      // büyük dosyalar için klasör
input string InpSmallFolderName="small"; // küçük dosyalar için klasör
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string   file_name;      // isimleri depolamak için bir değişken
   string   filter="*.csv"; // dosyaların aranması için filtre
   ulong    file_size=0;    // bayt bazında dosya büyüklüğü
   int      size=0;         // dosyaların sayısı
//--- çalışacağımız dosyanın adresini çıktıla
   PrintFormat("%s\\Files\\ klasöründe çalışılıyor",TerminalInfoString(TERMINAL_COMMON
//--- arama işleyicisini, tüm terminallerin ortak kök klasöründe al
   long search_handle=FileFindFirst(filter,file_name,FILE_COMMON);
//--- FileFindFirst() fonksiyonu başarıyla uygulanmış mı kontrol et
   if(search_handle!=INVALID_HANDLE)
     {
      //--- dosyaları boyutlarına göre döngü içinde taşı
      do
        {
         //--- dosyayı aç
         ResetLastError();
         int file_handle=FileOpen(file_name,FILE_READ|FILE_CSV|FILE_COMMON);
         if(file_handle!=INVALID_HANDLE)
```

```
            {
             //--- dosya boyutunu al
             file_size=FileSize(file_handle);
             //--- dosyayı kapa
             FileClose(file_handle);
            }
          else
            {
             PrintFormat("%s dosyası açılamadı, Hata kodu = %d",file_name,GetLastError
             continue;
            }
          //--- dosya boyutunu çıktıla
          PrintFormat("%s dosyasının boyutu %d bayttır",file_name,file_size);
          //--- dosyanın taşınacağı hedef adresi tanımla
          string path;
          if(file_size>InpThresholdSize*1024)
             path=InpBigFolderName+"//"+file_name;
          else
             path=InpSmallFolderName+"//"+file_name;
          //--- dosyayı taşı
          ResetLastError();
          if(FileMove(file_name,FILE_COMMON,path,FILE_REWRITE|FILE_COMMON))
             PrintFormat("%s dosyası taşındı",file_name);
          else
             PrintFormat("Hata, kod = %d",GetLastError());
         }
      while(FileFindNext(search_handle,file_name));
      //--- arama işleyicisini kapa
      FileFindClose(search_handle);
     }
   else
      Print("Dosyalar bulunamadı!");
  }
```

FileTell

Fonksiyon, açık bir dosyanın işaretçisinin mevcut konumuna dönüş yapar.

```
ulong  FileTell(
   int  file_handle    // Dosya tanıtıcısı
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

Dönüş değeri

Dosya işaretçisinin konumu - dosya başlangıcından itibaren, bayt bazında.

Not

Hata ile ilgili bilgi almak için GetLastError() fonksiyonunu çağırın.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                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"
//--- script çalıştırıldığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- giriş parametreleri
input string InpFileName="file.txt";     // dosya ismi
input string InpDirectoryName="Data";    // dizin ismi
input int    InpEncodingType=FILE_ANSI; // ANSI=32 veya UNICODE=64
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- rassal sayıları oluşturmak için değişken değerini belirle
   _RandomSeed=GetTickCount();
//--- dizgilerin başlangıç noktası konumları için değişkenler
   ulong pos[];
   int   size;
//--- hata değerini sıfırla
   ResetLastError();
//--- dosyayı aç
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_TXT|InpEr
   if(file_handle!=INVALID_HANDLE)
```

```
     {
      PrintFormat("%s dosyası, okuma için müsait",InpFileName);
      //--- dosyadaki her bir dizgi için başlangıç konumu al
      GetStringPositions(file_handle,pos);
      //--- dosyadaki dizgilerin sayısını tanımla
      size=ArraySize(pos);
      if(!size)
        {
         //--- dosyada dizgi yoksa dur
         PrintFormat("%s dosyası boş!",InpFileName);
         FileClose(file_handle);
         return;
        }
      //--- dizgi sayısını rassal olarak seç
      int ind=MathRand()%size;
      //--- konumu dizginin başlangıç noktasına kaydır
      FileSeek(file_handle,pos[ind],SEEK_SET);
      //--- ind sayısı ile dizgiyi oku ve çıktıla
      PrintFormat("%d numaralı dizgi metni: \"%s\"",ind,FileReadString(file_handle));
      //--- dosyayı kapat
      FileClose(file_handle);
      PrintFormat("%s dosyası kapatıldı",InpFileName);
     }
   else
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
  }
//+------------------------------------------------------------------------------------+
//| Bu fonksiyon, dosyadaki her dizgi için başlangıç konumları tanımlar                |
//| ve bunları arr dizisine yerleştirir                                                |
//+------------------------------------------------------------------------------------+
void GetStringPositions(const int handle,ulong &arr[])
  {
//--- varsayılan dizi büyüklüğü
   int def_size=127;
//--- dizi için bellek tahsis et
   ArrayResize(arr,def_size);
//--- dizgi sayacı
   int i=0;
//--- bu, dosyanın sonu değilse, hala en azından bir dizgi kalmıştır
   if(!FileIsEnding(handle))
     {
      arr[i]=FileTell(handle);
      i++;
     }
   else
      return; // dosya boş, çık
//--- kaydırmayı kodlama biçimine göre ve bayt bazında tanımla
   int shift;
   if(FileGetInteger(handle,FILE_IS_ANSI))
```

```
      shift=1;
   else
      shift=2;
//--- dizgileri döngü içinde incele
   while(1)
     {
      //--- dizgiyi oku
      FileReadString(handle);
      //--- dosya sonunu kontrol et
      if(!FileIsEnding(handle))
        {
         //--- bir sonraki dizginin konumunu kaydet
         arr[i]=FileTell(handle)+shift;
         i++;
         //--- taşma varsa dizi büyüklüğünü artır
         if(i==def_size)
           {
            def_size+=def_size+1;
            ArrayResize(arr,def_size);
           }
        }
      else
         break; // dosyanın sonuna gelindi, çık
     }
//--- dizinin gerçek büyüklüğünü tanımla
   ArrayResize(arr,i);
  }
```

FileWrite

Bu fonksiyon CSV dosyasına veri yazmak amacıyla tasarlanmıştır, ayırıcı - değeri 0 olmadığı sürece - otomatik olarak eklenecektir. Dosya yazımı tamamlandıktan sonra "\r\n" satır sonu karakteri eklenecektir.

```
uint  FileWrite(
   int  file_handle,    // Dosya işaretçisi
   ...                  // Kaydedilmiş parametrelerin listesi
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

...

[in] Dosyaya yazılacak parametrelerin listesi virgüllerle ayrılır. En fazla 63 adet parametre yazılabilir.

Dönüş değeri

Yazılmış baytların sayısı.

Not

Sayılar, çıktılama sırasında bir metne dönüştürülür (bakınız, Print() fonksiyonu). double tipi veriler, noktadan sonra 16 hanelik kesinliğe kadar, hangi notasyonun daha düzenli olduğuna bağlı olarak bilimsel veya geleneksel biçimlerde çıktılanabilir. float tipi veri, noktadan sonra 5 hane ile çıktılanır. Reel sayıları farklı çözünürlükle çıktılamak için, DoubleToString() fonksiyonunu kullanın.

bool tipli veriler "true" veya "false" dizgileriyle çıktılanır. datetime tipi veriler YYYY.MM.DD HH:MM:SS şeklinde görüntülenir.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                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"
//--- script çalıştığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- terminalden veri alımı için parametreler
input string             InpSymbolName="EURUSD";           // döviz çifti
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_H1;        // zaman aralığı
input int                InpFastEMAPeriod=12;              // hızlı EMA periyodu
input int                InpSlowEMAPeriod=26;              // yavaş EMA periyodu
input int                InpSignalPeriod=9;                // fark ortalama periyodu
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // fiyat tipi
```

```
input datetime           InpDateStart=D'2012.01.01 00:00'; // veri kopyalamanın başlay
//--- veriyi dosyaya yazmak için parametreler
input string             InpFileName="MACD.csv";  // dosya ismi
input string             InpDirectoryName="Data"; // dizin ismi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish; // veri kopyalama için son tarih
   bool     sign_buff[]; // sinyal dizisi (true - al, false - sat)
   datetime time_buff[]; // sinyalin geliş zamanı
   int      sign_size=0; // sinyal dizisi büyüklüğü
   double   macd_buff[]; // gösterge değerlerinin dizisi
   datetime date_buff[]; // gösterge tarihlerinin dizisi
   int      macd_size=0; // gösterge dizilerinin büyüklükleri
//--- son zaman, şimdiki zaman
   date_finish=TimeCurrent();
//--- MACD göstergesinin tanıtıcı değerini al
   ResetLastError();
   int macd_handle=iMACD(InpSymbolName,InpSymbolPeriod,InpFastEMAPeriod,InpSlowEMAPeri
   if(macd_handle==INVALID_HANDLE)
     {
      //--- gösterge tanıtıcı değeri alınamadı
      PrintFormat("Gösterge tanıtıcı değerinin alınması hatası. Hata kodu = %d",GetLas
      return;
     }
//--- gösterge değerleri hesaplanana kadar döngü içinde olunacak
   while(BarsCalculated(macd_handle)==-1)
      Sleep(10); // göstergenin tüm değerlerinin hesaplanabilmesi için durakla
//--- belli bir zaman aralığı için gösterge değerlerini kopyala
   ResetLastError();
   if(CopyBuffer(macd_handle,0,InpDateStart,date_finish,macd_buff)==-1)
     {
      PrintFormat("Gösterge değerlerinin kopyalanması başarısız. Hata kodu = %d",GetLa
      return;
     }
//--- gösterge değerleri için uygun zaman değerlerini kopyala
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
     {
      PrintFormat("Zaman değerleri kopyalanamadı. Hata kodu = %d",GetLastError());
      return;
     }
//--- serbest bellek, gösterge tarafından işgal edildi
   IndicatorRelease(macd_handle);
//--- tampon büyüklüğünü al
   macd_size=ArraySize(macd_buff);
//--- verileri analiz et ve gösterge sinyallerini diziye kopyala
```

```
   ArrayResize(sign_buff,macd_size-1);
   ArrayResize(time_buff,macd_size-1);
   for(int i=1;i<macd_size;i++)
     {
      //--- alım sinyali
      if(macd_buff[i-1]<0 && macd_buff[i]>=0)
        {
         sign_buff[sign_size]=true;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
      //--- satış sinyali
      if(macd_buff[i-1]>0 && macd_buff[i]<=0)
        {
         sign_buff[sign_size]=false;
         time_buff[sign_size]=date_buff[i];
         sign_size++;
        }
     }
//--- gösterge değerlerini yazmak için dosyayı aç (dosya yoksa, otomatik olarak oluştu
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası yazma için hazır",InpFileName);
      PrintFormat("Dosya yolu: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- önce, sinyal sayısını yaz
      FileWrite(file_handle,sign_size);
      //--- sinyallerin zamanlarını ve değerlerini dosyaya yaz
      for(int i=0;i<sign_size;i++)
         FileWrite(file_handle,time_buff[i],sign_buff[i]);
      //--- dosyayı kapat
      FileClose(file_handle);
      PrintFormat("Veri yazıldı, %s dosyası kapatıldı",InpFileName);
     }
   else
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
  }
```

Ayrıca Bakınız

Comment, Print, StringFormat

FileWriteArray

Bu fonksiyon string tipli olmayan herhangi bir diziyi BIN dosyasına yazar (dizgi ve dinamik dizi içermeyen bir yapı dizisi olabilir).

```
uint  FileWriteArray(
   int           file_handle,          // Dosya tanıtıcısı
   const void&   array[],              // Dizi
   int           start=0,              // Dizideki başlangıç indisi
   int           count=WHOLE_ARRAY     // Eleman sayısı
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

array[]

[out] Kayı yapılacak dizi.

start=0

[in] Dizideki başlangıç indisi (ilk kaydedilen elemanların sayısı).

count=WHOLE_ARRAY

[in] Kaydedilecek elemanların sayısı (WHOLE_ARRAY, start sayısından başlayarak dizinin sonuna kadar kayıt yapılacağını belirtir).

Dönüş değeri

Kaydedilen elemanların sayısı.

Not

Dizgilerden oluşan diziler TXT dosyasına kaydedilebilir. Bu durumda, dizgiler otomatik olarak "\r\n" satır sonu karakterleri ile sonlandırılır. Dosya tipine bağlı olarak ANSI veya UNICODE, dizgiler ansi-kodlamasına dönüştürülür yada dönüştürülmez.

Örnek:

```
//+------------------------------------------------------------------+
//|                                          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"
//--- giriş parametreleri
input string InpFileName="data.bin";
input string InpDirectoryName="SomeFolder";
//+------------------------------------------------------------------+
//| Fiyat verisini depolamak için bir yapı                           |
//+------------------------------------------------------------------+
```

```
struct prices
  {
   datetime          date; // tarih
   double            bid;  // satış fiyatı
   double            ask;  // alış fiyatı
  };
//--- global değişkenler
int    count=0;
int    size=20;
string path=InpDirectoryName+"//"+InpFileName;
prices arr[];
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizi için bellek tahsis et
   ArrayResize(arr,size);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- count<n ise kalan count dizgiyi yaz
   WriteData(count);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//--- veriyi diziye yaz
   arr[count].date=TimeCurrent();
   arr[count].bid=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   arr[count].ask=SymbolInfoDouble(Symbol(),SYMBOL_ASK);
//--- mevcut veriyi göster
         Print("Tarih = ",arr[i].date," Satış = ",arr[i].bid," Alış = ",arr[i].ask);
//--- sayacı artır
   count++;
//--- Dizi doldurulmuşsa, veriyi dosyaya yaz ve sıfırla
   if(count==size)
     {
      WriteData(size);
      count=0;
     }
  }
```

```
//+------------------------------------------------------------------+
//| Dizinin n elemanını dosyaya yaz                                  |
//+------------------------------------------------------------------+
void WriteData(const int n)
  {
//--- dosyayı aç
   ResetLastError();
   int handle=FileOpen(path,FILE_READ|FILE_WRITE|FILE_BIN);
   if(handle!=INVALID_HANDLE)
     {
      //--- dizideki veriyi dosyanın sonuna yaz
      FileSeek(handle,0,SEEK_END);
      FileWriteArray(handle,arr,0,n);
      //--- dosyayı kapat
      FileClose(handle);
     }
   else
      Print("Dosya açma başarısız, hata ",GetLastError());
  }
```

Ayrıca Bakınız

Değişkenler, FileSeek

FileWriteDouble

Bu fonksiyon, double tipli bir parametrenin değerini, dosya işaretçisinin mevcut konumundan başlayarak ikili bir dosyaya yazar.

```
uint  FileWriteDouble(
   int     file_handle,     // Dosya tanıtıcı değeri
   double  value            // Yazılacak değer
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

value

[in] double tipli değer.

Dönüş değeri

Başarılı sonuç durumunda fonksiyon, yazılan bayt sayısına dönüş yapar (bu durumda sizeof(double) =8). Ardından, dosya işaretçisi de eşit sayıda bayt ile kaydırılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                        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"
//--- script çalıştığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- terminalden veri alımı için parametreler
input string             InpSymbolName="EURJPY";           // döviz çifti
input ENUM_TIMEFRAMES    InpSymbolPeriod=PERIOD_M15;       // zaman aralığı
input int                InpMAPeriod=10;                   // düzleştirme periyodu
input int                InpMAShift=0;                     // gösterge kaydırma değeri
input ENUM_MA_METHOD     InpMAMethod=MODE_SMA;             // düzleştirme tipi
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;      // fiyat tipi
input datetime           InpDateStart=D'2013.01.01 00:00'; // veri kopyalama için baş
//--- verinin dosyaya yazılması için gereken parametreler
input string             InpFileName="MA.csv";     // dosya ismi
input string             InpDirectoryName="Data"; // dizin ismi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
```

```
   datetime date_finish=TimeCurrent();
   double   ma_buff[];
   datetime time_buff[];
   int      size;
//--- MA göstergesinin tanıtıcı değerini al
   ResetLastError();
   int ma_handle=iMA(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpMAShift,InpMAMethod,
   if(ma_handle==INVALID_HANDLE)
     {
      //--- gösterge tanıtıcı değeri alınamadı
      PrintFormat("Gösterge tanıtıcı değerinin alınması hatası. Hata kodu = %d",GetLas
      return;
     }
//--- gösterge değerleri hesaplanana kadar döngü içinde olunacak
   while(BarsCalculated(ma_handle)==-1)
      Sleep(20); // göstergenin değerlerinin hesaplanabilmesiiçin bir duraklama
   PrintFormat("gösterge değerleri, %s'den başlayarak dosyaya yazılacak",TimeToString
//--- gösterge değerlerini kopyala
   ResetLastError();
   if(CopyBuffer(ma_handle,0,InpDateStart,date_finish,ma_buff)==-1)
     {
      PrintFormat("Gösterge değerlerinin kopyalanması başarısız oldu. Hata kodu = %d",
      return;
     }
//--- uygun çubukların zamanlarını kopyala
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Zaman değerleri kopyalanamadı. Hata kodu = %d",GetLastError());
      return;
     }
//--- tampon büyüklüğünü al
   size=ArraySize(ma_buff);
//--- serbest bellek, gösterge tarafından işgal edildi
   IndicatorRelease(ma_handle);
//--- gösterge değerlerini yazmak için dosyayı aç (dosya yoksa, otomatik olarak oluştu
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası yazma için hazır",InpFileName);
      PrintFormat("Dosya yolu: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- önce, veri örneğinin büyüklüğünü yaz
      FileWriteDouble(file_handle,(double)size);
      //--- göstergenin zamanını ve değerini dosyaya yaz
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteDouble(file_handle,ma_buff[i]);
```

```
        }
      //--- dosyayı kapat
      FileClose(file_handle);
      PrintFormat("Veri yazıldı, %s dosyası kapatıldı",InpFileName);
     }
   else
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
  }
```

Ayrıca Bakınız

Reel tipler (double, float)

FileWriteFloat

Bu fonksiyon, float tipli bir parametrenin değerini, dosya işaretçisinin mevcut konumundan başlayarak ikili bir dosyaya yazar.

```
uint  FileWriteFloat(
   int    file_handle,     // Dosya tanıtıcısı
   float  value            // Yazılacak değer
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

value

[in] float tipli değer.

Dönüş değeri

Başarılı sonuç durumunda fonksiyon, yazılan bayt sayısına dönüş yapar (bu durumda sizeof(float)=4). Ardından, dosya işaretçisi de eşit sayıda bayt ile kaydırılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                          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"
//--- script çalıştığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- terminalden veri alımı için parametreler
input string          InpSymbolName="EURUSD";           // döviz çifti
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_M15;       // zaman aralığı
input datetime        InpDateStart=D'2013.01.01 00:00'; // veri kopyalamanın başlangıç
//--- verinin dosyaya yazılması için gereken parametreler
input string          InpFileName="Close.bin"; // dosya ismi
input string          InpDirectoryName="Data"; // dizin ismi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   double   close_buff[];
   datetime time_buff[];
   int      size;
```

```
//--- hata değerini sıfırla
   ResetLastError();
//--- her bir çubuğun kapanış fiyatını kopyala
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("Kapanış fiyatları kopyalanamadı. Hata kodu = %d",GetLastError());
      return;
     }
//--- her bir çubuk için zamanı kopyala
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Zaman değerleri kopyalanamadı. Hata kodu = %d",GetLastError());
      return;
     }
//--- tampon büyüklüğünü al
   size=ArraySize(close_buff);
//--- değerlerin yazılacağı dosyayı aç (dosya mevcut değilse, otomatik olarak oluştur
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası, yazma amacıyla açıldı",InpFileName);
      PrintFormat("Dosya yolu: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- kapanış fiyatı değerlerini ve zamanlarını yaz
      for(int i=0;i<size;i++)
        {
         FileWriteDouble(file_handle,(double)time_buff[i]);
         FileWriteFloat(file_handle,(float)close_buff[i]);
        }
      //--- dosyayı kapat
      FileClose(file_handle);
      PrintFormat("Veri yazıldı, %s dosyası kapatıldı",InpFileName);
     }
   else
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
  }
```

Ayrıca Bakınız

Reel tipler (double, float), FileWriteDouble

FileWriteInteger

Bu fonksiyon, int tipi bir parametrenin değerini, dosya işaretçisinin mevcut konumundan başlayarak ikili bir dosyaya yazar.

```
uint  FileWriteInteger(
   int  file_handle,         // Dosya tanıtıcı değeri
   int  value,               // Yazılacak değer
   int  size=INT_VALUE       // Bayt bazında boyut
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

value

[in] Tamsayı değeri.

size=INT_VALUE

[in] Yazılması gereken baytların sayısı (en fazla 4). Karşılık gelen sabitler verilmiştir: CHAR_VALUE = 1, SHORT_VALUE = 2 ve INT_VALUE = 4, böylece fonksiyon, char, uchar, short veya int tipli bir değeri yazabilir.

Dönüş değeri

Başarılı sonuç durumunda, yazılan bayt sayısına dönüş yapılır ve dosya işaretçisi, eşit sayıda bayt ile kaydırılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                      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"
//--- script çalıştığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- terminalden veri alımı için parametreler
input string              InpSymbolName="EURUSD";           // döviz çifti
input ENUM_TIMEFRAMES     InpSymbolPeriod=PERIOD_H1;        // zaman aralığı
input datetime            InpDateStart=D'2013.01.01 00:00'; // veri kopyalama için baş
//--- verinin dosyaya yazılması için gereken parametreler
input string              InpFileName="Trend.bin"; // dosya ismi
input string              InpDirectoryName="Data"; // dizin ismi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
```

```
  {
   datetime date_finish=TimeCurrent();
   double   close_buff[];
   datetime time_buff[];
   int      size;
//--- hata değerini sıfırla
   ResetLastError();
//--- her bir çubuğun kapanış fiyatını kopyala
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
      PrintFormat("Kapanış fiyatlarının değerleri kopyalanamadı. Hata kodu = %d",GetLa
      return;
     }
//--- her bir çubuk için zamanı kopyala
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Zaman değerleri kopyalanamadı. Hata kodu = %d",GetLastError());
      return;
     }
//--- tampon büyüklüğünü al
   size=ArraySize(close_buff);
//--- değerlerin yazılacağı dosyayı aç (dosya mevcut değilse, otomatik olarak oluştur
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası yazma için hazır",InpFileName);
      PrintFormat("Dosya yolu: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //---
      int   up_down=0; // trend bayrağı
      int   arr_size;  // arr dizisinin büyüklüğü
      uchar arr[];     // uchar tipi dizi
      //--- zaman değerlerini dosyaya yaz
      for(int i=0;i<size-1;i++)
        {
         //--- mevcut ve bir sonraki kapanış fiyatlarını karşılaştır
         if(close_buff[i]<=close_buff[i+1])
           {
            if(up_down!=1)
              {
               //--- FileWriteInteger fonksiyonu ile tarih değerini dosyaya yaz
               StringToCharArray(TimeToString(time_buff[i]),arr);
               arr_size=ArraySize(arr);
               //--- önce, dizideki sembol sayısını yaz
               FileWriteInteger(file_handle,arr_size,INT_VALUE);
               //--- sembolleri yaz
               for(int j=0;j<arr_size;j++)
                  FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
               //--- trend bayrağını değiştir
```

```
                up_down=1;
               }
            }
         else
            {
             if(up_down!=-1)
               {
                //--- FileWriteInteger kullanarak tarih değerini dosyaya yaz
                StringToCharArray(TimeToString(time_buff[i]),arr);
                arr_size=ArraySize(arr);
                //--- önce, dizideki sembol sayısını yaz
                FileWriteInteger(file_handle,arr_size,INT_VALUE);
                //--- sembolleri yaz
                for(int j=0;j<arr_size;j++)
                   FileWriteInteger(file_handle,arr[j],CHAR_VALUE);
                //--- trend bayrağını değiştir
                up_down=-1;
               }
            }
        }
      //--- dosyayı kapat
      FileClose(file_handle);
      PrintFormat("Veri yazıldı, %s dosyası kapatıldı",InpFileName);
     }
   else
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
  }
```

Ayrıca Bakınız

IntegerToString, StringToInteger, Tamsayı tipleri

FileWriteLong

Bu fonksiyon, long tipi bir parametrenin değerini, dosya işaretçisinin mevcut konumundan başlayarak ikili bir dosyaya yazar.

```
uint  FileWriteLong(
   int   file_handle,     // Dosya tanıtıcı değeri
   long  value            // Yazılacak değer
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

value

[in] long tipli değer.

Dönüş değeri

Başarılı sonuç durumunda fonksiyon, yazılan bayt sayısına dönüş yapar (bu durumda sizeof(long)=8). Ardından, dosya işaretçisi de eşit sayıda bayt ile kaydırılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                          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"
//--- script çalıştığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- terminalden veri alımı için parametreler
input string          InpSymbolName="EURUSD";           // döviz çifti
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;        // zaman aralığı
input datetime        InpDateStart=D'2013.01.01 00:00'; // veri kopyalamanın başlangıç
//--- verinin dosyaya yazılması için gereken parametreler
input string          InpFileName="Volume.bin"; // dosya ismi
input string          InpDirectoryName="Data";  // dizin (dosya konumu) ismi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish=TimeCurrent();
   long     volume_buff[];
   datetime time_buff[];
   int      size;
```

```
//--- hata değerini sıfırla
   ResetLastError();
//--- Her çubuk için tik değerlerini kopyala
   if(CopyTickVolume(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,volume_bu
     {
      PrintFormat("Tik hacmi değerleri kopyalanamadı. Hata kodu = %d",GetLastError());
      return;
     }
//--- her bir çubuk için zamanı kopyala
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Zaman değerleri kopyalanamadı. Hata kodu = %d",GetLastError());
      return;
     }
//--- tampon büyüklüğünü al
   size=ArraySize(volume_buff);
//--- gösterge değerlerini yazmak için dosyayı aç (dosya yoksa, otomatik olarak oluştu
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası yazma için hazır",InpFileName);
      PrintFormat("Dosya yolu: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- önce, veri örnek büyüklüğünü yaz
      FileWriteLong(file_handle,(long)size);
      //--- zaman ve hacim değerlerini dosyaya yaz
      for(int i=0;i<size;i++)
        {
         FileWriteLong(file_handle,(long)time_buff[i]);
         FileWriteLong(file_handle,volume_buff[i]);
        }
      //--- dosyayı kapat
      FileClose(file_handle);
      PrintFormat("Veri yazıldı, %s dosyası kapatıldı",InpFileName);
     }
   else
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
  }
```

Ayrıca Bakınız

Tamsayı tipleri, FileWriteInteger

FileWriteString

Bu dosya, bir BIN veya TXT dosyasının içine, dosya işaretçisinin mevcut konumundan başlayarak string tipi bir değer yazar. CSV veya TXT dosyası yazarken; dizgi içinde '\r' (CR) sembolü olmadan yazılmış bir '\n' (LF) sembolü bulunuyorsa, '\n' karakterinden önce '\r' eklenir.

```
uint  FileWriteString(
   int           file_handle,     // Dosya tanıtıcısı
   const string  text_string,     // yazılacak dizgi
   int           length=-1        // sembollerin sayısı
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

text_string

[in] Dizgi.

length=-1

[in] Yazılacak karakterlerin sayısı. Bu seçenek, bir BIN dosyasına dizgi yazılırken gereklidir. Boyut belirtilmediği takdirde, 0 taşıyıcısı olmadan tüm dizgi yazılır. Dizginin büyüklüğünden daha küçük bir boyut belirtilmişse, 0 taşıyıcısı olmadan dizginin bir bölümü yazılır. Dizgi büyüklüğünden daha büyük bir boyut belirtilmişse, dizginin kalan bölümü uygun sayıda sıfır ile doldurulur. CSV ve TXT tipi dosyalar için, bu parametre gözardı edilir ve dizginin bütünü yazılır.

Dönüş değeri

Başarılı sonuç durumunda, yazılan bayt sayısına dönüş yapılır ve dosya işaretçisi, eşit sayıda bayt ile kaydırılır.

Not

FILE_UNICODE bayrağı ile (veya FILE_ANSI bayrağı olmadan) bir dosya açarken, yazılacak bayt sayısının dizgi karakterlerinin iki katına çıkacağını not edin. FILE_ANSI bayrağı ile açılmış bir dosyaya yazım yapılırken; yazılan baytların sayısı, dizgi karakterlerinin sayısı ile örtüşür.

Örnek:

```
//+------------------------------------------------------------------+
//|                                        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"
//--- script çalıştığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- terminalden veri alımı için parametreler
input string              InpSymbolName="EURUSD";           // döviz çifti
input ENUM_TIMEFRAMES     InpSymbolPeriod=PERIOD_H1;        // zaman aralığı
```

```
input int                InpMAPeriod=14;                  // MA periyodu
input ENUM_APPLIED_PRICE InpAppliedPrice=PRICE_CLOSE;     // fiyat tipi
input datetime           InpDateStart=D'2013.01.01 00:00'; // veri kopyalama için baş
//--- verinin dosyaya yazılması için gereken parametreler
input string             InpFileName="RSI.csv";   // dosya ismi
input string             InpDirectoryName="Data"; // dizin ismi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime date_finish; // veri kopyalama için son tarih
   double   rsi_buff[];  // gösterge değerlerinin dizisi
   datetime date_buff[]; // göstege tarihlerinin dizisi
   int      rsi_size=0;  // gösterge dizilerinin büyüklüğü
//--- durdurma zamanı şu anki zaman olsun
   date_finish=TimeCurrent();
//--- RSI göstergesinin tanıtıcı değerini al
   ResetLastError();
   int rsi_handle=iRSI(InpSymbolName,InpSymbolPeriod,InpMAPeriod,InpAppliedPrice);
   if(rsi_handle==INVALID_HANDLE)
     {
      //--- gösterge tanıtıcı değeri alınamadı
      PrintFormat("Gösterge tanıtıcı değerinin alınması hatası. Hata kodu = %d",GetLas
      return;
     }
//--- gösterge, verileri hesaplayana kadar döngüde kalacak
   while(BarsCalculated(rsi_handle)==-1)
      Sleep(10); // gösterge değerlerinin hesaplanması için bir duraklama
//--- belli bir zaman aralığı için gösterge değerlerini kopyala
   ResetLastError();
   if(CopyBuffer(rsi_handle,0,InpDateStart,date_finish,rsi_buff)==-1)
     {
      PrintFormat("Gösterge değerlerinin kopyalanması başarısız. Hata kodu = %d",GetLa
      return;
     }
//--- gösterge değerleri için uygun zaman değerlerini kopyala
   ResetLastError();
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,date_buff)==-1)
     {
      PrintFormat("Zaman değerleri kopyalanamadı. Hata kodu = %d",GetLastError());
      return;
     }
//--- serbest bellek, gösterge tarafından işgal edildi
   IndicatorRelease(rsi_handle);
//--- tampon büyüklüğünü al
   rsi_size=ArraySize(rsi_buff);
//--- gösterge değerlerini yazmak için dosyayı aç (dosya yoksa, otomatik olarak oluştu
   ResetLastError();
```

```
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası yazma için hazır",InpFileName);
      PrintFormat("Dosya yolu: %s\\Files\\",TerminalInfoString(TERMINAL_DATA_PATH));
      //--- ek değişkenleri hazırla
      string str="";
      bool   is_formed=false;
      //--- aşırı-alım ve aşırı-satım alanlarının oluşum tarihlerini yaz
      for(int i=0;i<rsi_size;i++)
        {
         //--- gösterge değerlerini kontrol et
         if(rsi_buff[i]>=70 || rsi_buff[i]<=30)
           {
            //--- eğer değer, bu alandaki ilk değer ise
            if(!is_formed)
              {
               //--- tarihi ve değeri ekle
               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];
            //--- yeni döngü tekrarına geç
            continue;
           }
         //--- bayrağı kontrol et
         if(is_formed)
           {
            //--- dizgi şekillendi, dosyaya yaz
            FileWriteString(file_handle,str+"\r\n");
            is_formed=false;
           }
        }
      //--- dosyayı kapat
      FileClose(file_handle);
      PrintFormat("Veri yazıldı, %s dosyası kapatıldı",InpFileName);
     }
   else
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
  }
```

Ayrıca Bakınız

string Tipi, StringFormat

FileWriteStruct

Bu fonksiyon, parametre olarak geçirilmiş bir yapının içeriğini, dosya işaretçisinin mevcut konumundan itibaren bir bin-dosyasına yazar

```
uint  FileWriteStruct(
   int           file_handle,       // Dosya tanıtıcı değeri
   const void&   struct_object,     // bir nesneye yapılan referans
   int           size=-1            // bayt bazında, yazılacak büyüklük
   );
```

Parametreler

file_handle

[in] FileOpen() fonksiyonunun dönüş yaptığı dosya tanımlayıcısı.

struct_object

[in] Yapı nesnesine yapılan referans. Bu yapı, dizgiler, dinamik diziler veya sanal fonksiyonlar içermemelidir.

size=-1

[in] Yazılacak bayt sayısı. Boyut belirtilmemişse veya belirtilen değer yapının boyutunu aşıyorsa, söz konusu yapının boyutu kullanılır.

Dönüş değeri

Başarılı sonuç durumunda, yazılan bayt sayısına dönüş yapılır ve dosya işaretçisi, eşit sayıda bayt ile kaydırılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                          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"
//--- script çalıştığında giriş parametrelerinin penceresini göster
#property script_show_inputs
//--- terminalden veri alımı için parametreler
input string          InpSymbolName="EURUSD";           // döviz çifti
input ENUM_TIMEFRAMES InpSymbolPeriod=PERIOD_H1;        // zaman aralığı
input datetime        InpDateStart=D'2013.01.01 00:00'; // veri kopyalamanın başlangıc
//--- verinin dosyaya yazılması için gereken parametreler
input string          InpFileName="EURUSD.txt";         // dosya ismi
input string          InpDirectoryName="Data";          // dizin ismi
//+------------------------------------------------------------------+
//| Mum verilerini depolamak için bir yapı                           |
//+------------------------------------------------------------------+
struct candlesticks
```

```
  {
   double            open;  // açılış fiyatı
   double            close; // kapanış fiyatı
   double            high;  // yüksek fiyat
   double            low;   // düşük fiyat
   datetime          date;  // tarih
  };
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   datetime     date_finish=TimeCurrent();
   int          size;
   datetime     time_buff[];
   double       open_buff[];
   double       close_buff[];
   double       high_buff[];
   double       low_buff[];
   candlesticks cand_buff[];
//--- hata değerini sıfırla
   ResetLastError();
//--- çubukların açılış zamanı değerlerini kopyala
   if(CopyTime(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,time_buff)==-1)
     {
      PrintFormat("Zaman değerleri kopyalanamadı. Hata kodu = %d",GetLastError());
      return;
     }
//--- çubukların yüksek fiyatlarını al
   if(CopyHigh(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,high_buff)==-1)
     {
      PrintFormat("High (yüksek) fiyat değerleri kopyalanamadı. Hata kodu = %d",GetLas
      return;
     }
//--- çubukların düşük fiyatlarını al
   if(CopyLow(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,low_buff)==-1)
     {
      PrintFormat("Low (düşük) fiyat değerleri kopyalanamadı. Hata kodu = %d",GetLastE
      return;
     }
//--- çubukların açılış fiyatı değerlerini al
   if(CopyOpen(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,open_buff)==-1)
     {
      PrintFormat("Open (açılış) fiyatı değerleri alınamadı. Hata kodu = %d",GetLastEr
      return;
     }
//--- çubukların kapanış fiyatı değerlerini al
   if(CopyClose(InpSymbolName,InpSymbolPeriod,InpDateStart,date_finish,close_buff)==-1
     {
```

```
      PrintFormat("Close (kapanış) fiyatı değerleri alınamadı. Hata kodu = %d",GetLast
      return;
     }
//--- dizilerin boyutlarını tanımla
   size=ArraySize(time_buff);
//--- yapı dizisindeki tüm veriyi kaydet
   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];
     }

//--- yapı dizisinin yazılacağı dosyayı aç (dosya mevcut değilse, otomatik olarak olus
   ResetLastError();
   int file_handle=FileOpen(InpDirectoryName+"//"+InpFileName,FILE_READ|FILE_WRITE|FI
   if(file_handle!=INVALID_HANDLE)
     {
      PrintFormat("%s dosyası, yazma amacıyla açıldı",InpFileName);
      PrintFormat("Dosya adresi: %s\\Files\\",TerminalInfoString(TERMINAL_COMMONDATA_
      //--- bayt sayısı için sayacı hazırla
      uint counter=0;
      //--- dizi değerlerini döngüye yaz
      for(int i=0;i<size;i++)
         counter+=FileWriteStruct(file_handle,cand_buff[i]);
      PrintFormat("%d baytlık bilgi, %s dosyasına yazıldı",InpFileName,counter);
      PrintFormat("Toplam bayt sayısı: %d * %d * %d = %d, %s",size,5,8,size*5*8,size*
      //--- dosyayı kapat
      FileClose(file_handle);
      PrintFormat("Veri yazıldı, %s dosyası kapatıldı",InpFileName);
     }
   else
      PrintFormat("%s dosyası açılamadı, Hata kodu = %d",InpFileName,GetLastError());
  }
```

Ayrıca Bakınız

Yapılar ve sınıflar

FileLoad

Belirtilen ikili dosyadaki tüm verileri parametre şeklinde geçirilen sayısal diziye yazar. Bu fonksiyon sayesinde, bilinen veri tiplerini uygun bir diziye kolayca yazabilirsiniz.

```
long  FileLoad(
   const string  file_name,          // Dosya ismi
   void&         buffer[],           // Bir sayısal tipli dizi veya basit yapı
   int           common_flag=0       // Dosya bayrağı, varsayılan olarak dosyalar <veri
   );
```

Parametreler

file_name

[in] Verinin okunacağı dosyanın ismi.

buffer

[out] Bir sayısal dizi veya basit yapı.

common_flag=0

[in] Dosya bayrağı (işlem modunu belirtir). Parametre belirtilmemişse dosya MQL5\Files (veya sınama için <sınama_temsilcisi_konumu>\MQL5\Files) konumunda aranır.

Dönüş Değeri

Okunan veri sayısı veya hata durumunda -1.

Not

FileLoad() fonksiyonu dosyadan sadece dizi büyklüğünün katlarına karşılık gelen bayt sayısını okur. Dosya boyutu 10 bayt olsun ve fonksiyon verileri double tipli bir diziye yazacak olsun (sizeof(double) =8). Bu durumda fonksiyon dosyadan sadece 8 baytlık veri okur. Kalan 2 bayt okunmaz ve FileLoad() fonksiyonu 1 (1 eleman okundu) dönüşü yapar.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                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
//--- giriş parametreleri
input int      bars_to_save=10; // Çubukların sayısı
//+------------------------------------------------------------------+
//| Script program start function                                    |
```

```
//+------------------------------------------------------------------+
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) ile %d çubuk kopyalandı",_Symbol,copied);
      //---  Fiyatların dosyaya yazılması
      if(!FileSave(filename,rates,FILE_COMMON))
         PrintFormat("FileSave() başarısız oldu, hata=%d",GetLastError());
     }
   else
      PrintFormat("CopyRates(%s) başarısız oldu, hata=%d",_Symbol,GetLastError());
//--- Şimdi fiyatları dosyadan okutalım
   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);
        }
     }
  }
```

Ayrıca bakınız

Yapılar ve Sınıflar, FileReadArray, FileReadStruct, FileSave

FileSave

Parametre şeklinde geçirilen dizinin tüm elemanlarını bir ikili dosyaya yazar. Bu fonksiyon sayesinde sayısal tipli dizileri ve basit yapıları tek bir dizgi şeklinde kolayca dosyaya yazabilirsiniz.

```
bool  FileSave(
   const string  file_name,          // Dosya ismi
   void&         buffer[],           // Bir sayısal tipli dizi veya basit yapı
   int           common_flag=0       // Dosya bayrağı, varsayılan olarak dosyalar <veri
   );
```

Parametreler

file_name

[in] Dizinin yazılacağı dosyanın ismi.

buffer

[in] Bir sayısal dizi veya basit yapı.

common_flag=0

[in] Dosya bayrağı (işlem modunu belirtir). Parametre belirtilmemişse dosya MQL5\Files (veya sınama için <sınama_temsilcisi_konumu>\MQL5\Files) konumuna yazılır.

Dönüş Değeri

Başarısızlık durumunda 'false' dönüşü yapar.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                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
//--- giriş parametreleri
input int      ticks_to_save=1000; // Tiklerin sayısı
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string  filename=_Symbol+"_ticks.bin";
   MqlTick ticks[];
u//---
   int copied=CopyTicks(_Symbol,ticks,COPY_TICKS_ALL,0,ticks_to_save);
   if(copied!=-1)
     {
      PrintFormat(" CopyTicks(%s) ile %d tik kopyaladı",_Symbol,copied);
```

```
      //--- tik geçmişi eşitlenmişse hata kodu sıfıra eşittir
      if(!GetLastError()==0)
         PrintFormat("%s: Tikler eşitlenmedi, hata=%d",_Symbol,copied,_LastError);
      //---  Tikler dosyaya yazılıyor
      if(!FileSave(filename,ticks,FILE_COMMON))
         PrintFormat("FileSave() başarısız oldu, hata=%d",GetLastError());
     }
   else
      PrintFormat("CopyTicks(%s) başarısız oldu, hata=%d",_Symbol,GetLastError());
//--- Şimdi dosyadan tikleri okuyalım
   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);
        }
     }
  }
```

Ayrıca bakınız

Yapılar ve Sınıflar, FileWriteArray, FileWriteStruct, FileLoad, FileWrite

FolderCreate

Files dizininde (common_flag değerine bağlı olarak) bir klasör oluşturur.

```
bool  FolderCreate(
   string  folder_name,        // Oluşturulacak yeni klasörün ismi
   int     common_flag=0       // Kapsam
   );
```

Parametreler

folder_name

[in] Oluşturulacak yeni klasörün ismi. Contains the relative path to the folder.

common_flag=0

[in] Dizin konumunu belirleyen bayrak. common_flag = FILE_COMMON ise, klasör tüm müşteri terminallerinin ortak klasöründe bulunur \Terminal\Common\Files. Aksi durumda, yerel bir klasördedir (sınama durumunda MQL5\Files veya MQL5\Tester\Files).

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Not

Dosya işlemleri, MQL5 dili içerisinde güvenlik amacıyla sıkı şekilde kontrol edilmektedir. İşlemleri, MQL5 araçları ile yürütülen dosyalar, dosya güvenlik-ortamı (sandbox) dışında yer alamaz.

Örnek:

```
#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;
```

```
}
```

Ayrıca Bakınız

FileOpen(), FolderClean(), FileCopy()

FolderDelete

Belirtilen klasörü siler. Klasör boş değilse, işlem gerçekleştirilemez.

```
bool  FolderDelete(
   string  folder_name,        // Silinecek klasörün ismi
   int     common_flag=0       // Kapsam
   );
```

Parametreler

folder_name

[in] Silinmesi istenen klasörün ismi. Klasörün tam adresini içerir.

common_flag=0

[in] Dizin konumunu belirleyen bayrak. common_flag = FILE_COMMON ise, klasör tüm müşteri terminallerinin ortak klasöründe bulunur \Terminal\Common\Files. Aksi durumda, yerel bir klasördedir (sınama durumunda MQL5\Files veya MQL5\Tester\Files).

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Not

Dosya işlemleri, MQL5 dili içerisinde güvenlik amacıyla sıkı şekilde kontrol edilmektedir. İşlemleri, MQL5 araçları ile yürütülen dosyalar, dosya güvenlik-ortamı (sandbox) dışında yer alamaz.

Eğer klasörde en azından bir dosya ve/veya alt-klasör bulunuyorsa, o zaman klasör silinmeden önce boşaltılmalıdır, aksi durumda silinemez. FolderClean() fonksiyonu, bir klasörün tüm içeriğini silmek için kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                            Demo_FolderDelete.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- Açıklama
#property description "Bu script, FolderDelete() fonksiyonunun kullanım örneğini göste
#property description "İlk olarak, iki klasör oluşturur, bunlardan biri boştur, ikinci
#property description "Boş olmayan bir klasörü silmeye çalıştığımızda hata dönüşü yapa

//--- Giriş parametrelerinin iletişim penceresini, script çalıştırıldığında göster
#property script_show_inputs
//--- Giriş parametreleri
input string   firstFolder="empty";     // Bir boş klasör
input string   secondFolder="nonempty";// dosyanın oluşturulacağı klasör
string filename="delete_me.txt";        // secondFolder içinde oluşturulacak dosyanın
```

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- Dosya tanıtıcısını buraya yaz
   int handle;
//--- Hangi dosyayı aradığımızı öğren
   string working_folder=TerminalInfoString(TERMINAL_DATA_PATH)+"\\MQL5\\Files";
//--- Bir hata ayıklama mesajı
   PrintFormat("working_folder=%s",working_folder);
//--- MQL5\Files adresinde boş bir klasör oluşturmayı deniyor
   if(FolderCreate(firstFolder,0)) // 0 değeri, terminalin yerel klasöründe çalıştığın
     {
      //--- Tam adresi, oluşturulan klasöre gir
      PrintFormat("%s klasörü oluşturuldu",working_folder+"\\"+firstFolder);
      //--- Hata kodunu sıfırla
      ResetLastError();
     }
   else
      PrintFormat("%s klasörü oluşturulamadı. Hata kodu %d",working_folder+"\\"+firstF

//--- şimdi, FileOpen() fonksiyonunu kullanarak, boş olmayan bir klasör yarat
   string filepath=secondFolder+"\\"+filename;  // Olmayan bir klasör içinde, yazma ar
   handle=FileOpen(filepath,FILE_WRITE|FILE_TXT); // FILE_WRITE bayrağı bu durumda zor
   if(handle!=INVALID_HANDLE)
      PrintFormat("%s dosyası, okuma amacıyla açıldı",working_folder+"\\"+filepath);
   else
      PrintFormat("%s dosyası, %s klasöründe oluşturulamadı. Hata kodu=",filename,secc

   Comment(StringFormat("Prepare to delete folders %s and %s", firstFolder, secondFold
//--- Çizelgedeki mesajı okumak için 5 saniyelik kısa bir duraklama
   Sleep(5000); // Sleep(), göstergelerde kullanılamaz!

//--- İletişim kutusu göster ve kullanıcıya sor
   int choice=MessageBox(StringFormat("%s ve %s klasörlerini silmek istiyor musunuz?",
                         "Klasörler siliniyor",
                         MB_YESNO|MB_ICONQUESTION); //  İki düğme - "Yes" ve "No"

//--- Seçilen versiyona göre bir eylem gerçekleştir
   if(choice==IDYES)
     {
      //--- Çizelgeden yorumu sil
      Comment("");
      //--- "Experts" (Uzmanlar) bültenine bir mesaj ekle
      PrintFormat("%s ve %s klasörleri siliniyor",firstFolder, secondFolder);
      ResetLastError();
      //--- Boş klasörleri sil
      if(FolderDelete(firstFolder))
         //--- Klasör boş olduğunda, şu mesajın gösterilmesi gerek
```

```
         PrintFormat("%s klasörü başarıyla silindi",firstFolder);
      else
         PrintFormat("%s klasörü silinemedi. Hata kodu=%d", firstFolder, GetLastError

      ResetLastError();
      //--- Bir dosya içeren klasörü sil
      if(FolderDelete(secondFolder))
         PrintFormat("%s klasörü başarıyla silindi", secondFolder);
      else
         //--- Klasör dosya içeriyorsa, şu mesajın gösterilmesi gerekir
         PrintFormat("%s klasörü silinemedi. Hata kodu=%d", secondFolder, GetLastErro
     }
   else
      Print("Silme işlemi iptal edildi");
//---
  }
```

Ayrıca Bakınız

FileOpen(), FolderClean(), FileMove()

FolderClean

Belirtilen klasördeki tüm dosyaları siler.

```
bool  FolderClean(
   string  folder_name,       // Silinecek dosyanın ismi
   int     common_flag=0      // Kapsam
   );
```

Parametreler

folder_name

[in] İçindeki tüm dosyaların silinmesi istenen klasörün ismi. Klasörün tam adresini içerir.

common_flag=0

[in] Dizin konumunu belirleyen bayrak. common_flag = FILE_COMMON ise, tüm müşteri terminallerinin ortak klasöründe bulunur \Terminal\Common\Files. Aksi durumda dizin, yerel bir klasördedir (sınama durumunda MQL5\Files veya MQL5\Tester\Files).

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Not

Dosya işlemleri, MQL5 dili içerisinde güvenlik amacıyla sıkı şekilde kontrol edilmektedir. İşlemleri, MQL5 araçları ile yürütülen dosyalar, dosya güvenlik-ortamı (sandbox) dışında yer alamaz.

Tüm dosya ve alt klasörler geri dönülemez biçimde silineceğinden, fonksiyon dikkatle kullanılmalıdır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                              Demo_FolderClean.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                               https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
//--- Açıklama
#property description "Bu script, FolderClean() fonksiyonunun kullanım örneğini göste
#property description "Önce, belirtilen klasörde FileOpen() fonksiyonu ile dosyalar o
#property description "Ardından dosyalar silinmeden önce MessageBox() fonksiyonu ile k

//--- giriş parametrelerinin iletişim penceresini, script çalıştırıldığında göster
#property script_show_inputs
//--- Giriş parametreleri
input string   foldername="demo_folder";  // MQL5/Files/ dizininde bir klasör oluştur
input int      files=5;                   // Oluşturulacak ve silinecek dosyaların say
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

```
void OnStart()
  {
   string name="testfile";
//--- Dosyaları, önce terminal veri klasöründe aç veya oluştur
   for(int N=0;N<files;N++)
     {
      //--- 'demo_folder\testfileN.txt' biçimde dosya ismi
      string filemane=StringFormat("%s\\%s%d.txt",foldername,name,N);
      //--- Yazma bayrağı ile bir dosya aç, bu durumda otomatik olarak 'demo_folder' k
      int handle=FileOpen(filemane,FILE_WRITE);
      //--- FileOpen() fonksiyonu başarılı oldu mu öğren
      if(handle==INVALID_HANDLE)
        {
         PrintFormat("%s dosyasının oluşturulması başarısız. Hata kodu",filemane,GetLa
         ResetLastError();
        }
      else
        {
         PrintFormat("%s dosyası, başarılı biçimde açıldı",filemane);
         //--- Açılan dosyaya artık ihtiyaç yok; dosyayı kapat
         FileClose(handle);
        }
     }

//--- Klasördeki dosyaların sayısını kontrol et
   int k=FilesInFolder(foldername+"\\*.*",0);
   PrintFormat("%s klasörü toplamda %d adet dosya içeriyor",foldername,k);

//--- Kullanıcıya sormak için bir iletişim penceresi aç
   int choice=MessageBox(StringFormat("%s klasöründen %d adet dosya sileceksiniz. Deva
                         "Dosyalar klasörden siliniyor",
                         MB_YESNO|MB_ICONQUESTION); //  İki düğme - "Yes" ve "No"
   ResetLastError();

//--- Seçilen versiyona göre bir eylem gerçekleştir
   if(choice==IDYES)
     {
      //--- Dosyaları silmeye başla
      PrintFormat("%s klasöründeki tüm dosyaların silinmesi için uğraşılıyor",folderna
      if(FolderClean(foldername,0))
         PrintFormat("Dosyalar başarıyla silindi, %d adet dosya, %s klasöründe kaldı",
                     foldername,
                     FilesInFolder(foldername+"\\*.*",0));
      else
         PrintFormat("%s klasöründen dosyalar silinemedi. Hata kodu %d",foldername,Get
     }
   else
      PrintFormat("Silme işlemi iptal edildi");
//---
  }
```

```
//+------------------------------------------------------------------+
//| Belirtilen klasördeki dosya sayısına dönüş yapar                 |
//+------------------------------------------------------------------+
int FilesInFolder(string path,int flag)
  {
   int count=0;
   long handle;
   string filename;
//---
   handle=FileFindFirst(path,filename,flag);
//--- Bir dosya bile bulunmuşsa, daha fazlası için aramaya devam et
   if(handle!=INVALID_HANDLE)
     {
      //--- Dosya ismini göster
      PrintFormat("%s dosyası bulundu",filename);
      //--- Bulunan dosyaların/klasörlerin sayacını artır
      count++;
      //--- dosyaları/klasörleri aramaya başla
      while(FileFindNext(handle,filename))
        {
         PrintFormat("%s dosyası bulundu",filename);
         count++;
        }
      //--- Tamamlanmasının ardından arama işleyicisini kapamayı unutma
      FileFindClose(handle);
     }
   else // İşleyici alınamadı
     {
      PrintFormat("%s klasöründeki dosya araması başarısız oldu",path);
     }
//--- sonuca dönüş yap
   return count;
  }
```

Ayrıca Bakınız

FileFindFirst, FileFindNext, FileFindClose

Özel Göstergeler

Özel göstergelerin tasarımında kullanılan grup fonksiyonları. Bu fonksiyonlar, Uzman Danışmanların ve script dosyalarının yazımında kullanılamaz.

Fonksiyon	Eylem
SetIndexBuffer	Belirtilen gösterge tamponunu, tek boyutlu, double tipli dinamik bir diziye bağlar
IndicatorSetDouble	double tipi gösterge özelliğinin değerini ayarlar
IndicatorSetInteger	int tipi gösterge özelliğinin değerini ayarlar
IndicatorSetString	string tipi gösterge özelliğinin değerini ayarlar
PlotIndexSetDouble	double tipi gösterge çizgisi özelliğinin değerini ayarlar
PlotIndexSetInteger	int tipi gösterge çizgisi özelliğinin değerini ayarlar
PlotIndexSetString	string tipi gösterge çizgisi özelliğinin değerini ayarlar
PlotIndexGetInteger	tam-sayı tipli gösterge çizgisi özelliğinin değerine dönüş yapar

Gösterge özellikleri, derleyici direktiflerini veya fonksiyonları kullanılarak ayarlanabilir. Bunu daha iyi anlamanız için, örneklerdeki gösterge stilleri üzerinde çalışmanız tavsiye edilir.

Bir özel gösterge için gereken tüm hesaplamalar, ön tanımlı OnCalculate() fonksiyonu içinde yer almalıdır. Eğer, OnCalculate() fonksiyonunun kısa biçimdeki çağrısını kullanıyorsanız

```
int OnCalculate (const int rates_total,      // price[] dizisinin büyüklüğü
```

burada *rates_total* değişkeni, gösterge değerlerinin hesaplanması için parametre olarak fonksiyona geçirilen price[] dizisindeki, toplam eleman sayısını belirtir.

prev_calculated parametresi, OnCalculate() fonksiyonunun bir önceki çağrıdaki uygulama sonucudur; gösterge değerlerinin hesaplanması için algoritmanın düzenlenmesini ve saklanmasını sağlar. Örneğin mevcut değerle rates_total = 1000, prev_calculated = 999 ise, her gösterge tamponunun sadece bir değeri için hesaplama yapmamız yeterli olabilir.

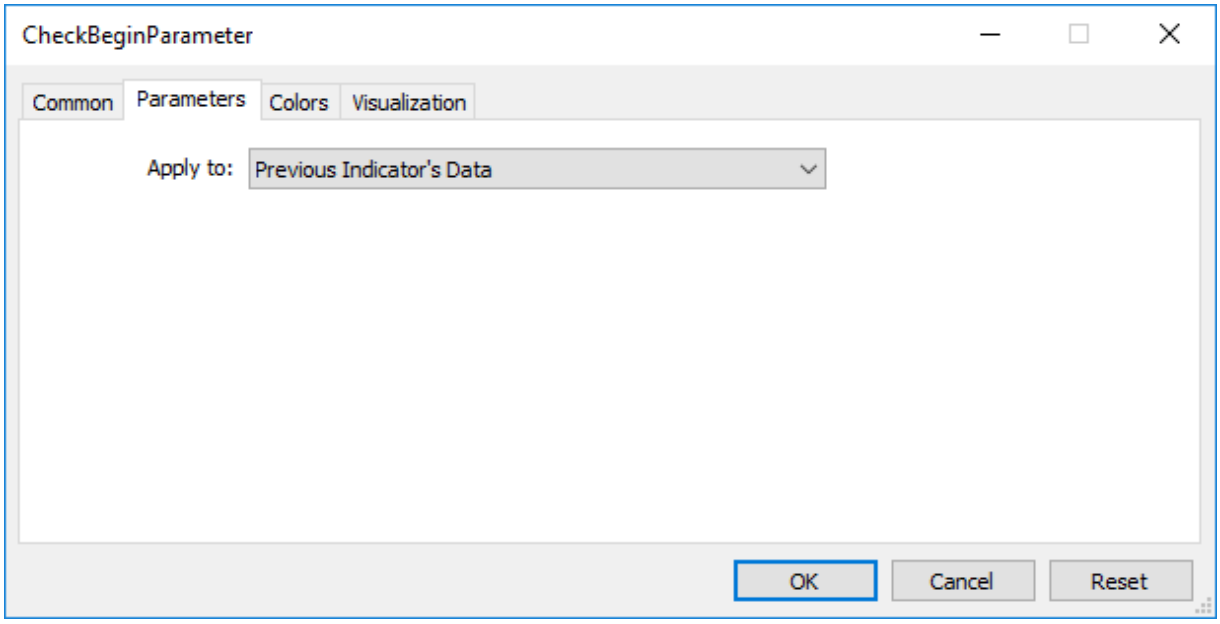
Eğer "price" dizisinin büyüklüğü ile ilgili bilgi mevcut olmasaydı, bu durumda, her bir gösterge tamponu için 1000 değerin hesaplanması gerekliliği ortaya çıkarırdı. OnCalculate() fonksiyonunun ilk çağrısında, prev_calculated = 0 olarak verilir. Eğer price[] dizisi bir şekilde değişmişse, bu durumda prev_calculated değeri 0'a eşit olur.

begin parametresi, price dizisindeki başlangıç değerlerinin sayısını verir; hesaplanabilir veri içermez. Örneğin, Accelerator Oscillator (ilk 37 değerin hesaplanmadığı) göstergesinin değerleri, giriş parametresi olarak kullanılırsa, begin = 37 olur. Örnek olarak, basit bir göstergeyi düşünelim:

```
#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
//--- gösterge tamponları
double         Label1Buffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Bunu "Klavuz" penceresinden Accelerator Oscillator göstergesinin penceresine sürükleyelim ve hesaplamaların, mevcut (önceki) bir göstergenin değerleriyle yapılacağını belirtelim:



Sonuç olarak prev_calculated değeri, OnCalculate() fonksiyonunun ilk çağrısında sıfıra, daha sonraki çağrılarda ise (fiyat çizelgesinde bulunan çubukların sayısı değişene kadar) rates_total değerine eşit olacaktır.

'begin' parametresinin değeri tam olarak başlangıçtaki çubuklarının sayısına eşittir ve Accelerator göstergesinin mantığına göre bu çubukların değerleri hesaplamaya katılmayacaktır. Accelerator.mq5 özel göstergesinin kaynak koduna bakacak olursak, OnInit() fonksiyonundaki şu satırları görürüz:

```
//--- çizime hangi çubuğun indisinden başlanacağını ayarlar
   PlotIndexSetInteger(0,PLOT_DRAW_BEGIN,37);
```

PlotIndexSetInteger(0, PLOT_DRAW_BEGIN, empty_first_values) fonksiyonunu kullanarak, gösterge tamponunun sıfır indisli dizisindeki var olmayan, (empty_first_values) yani hesaplamaya katılmayacak ilk değerlerin sayısını ayarlarız. Bu şekilde, aşağıdaki mekanizmalara sahip oluruz:

1. başka bir göstergenin hesaplanmasında kullanılmaması gereken başlangıç değerlerinin sayısının ayarlanması;
2. başka bir özel göstergenin çağrılması durumunda, gözardı edilecek ilk değerlerin sayısına dair bilginin edinilmesi.

Örneklerle Gösterge Stilleri

MetaTrader 5 Müşteri Terminali, uygun fonksiyonlar ile MQL5 uygulamalarında kullanılabilecek 38 adet teknik gösterge içerir. Ama MQL5 dilinin asıl avantajı, Uzman Danışmanlarda veya çizelgeye eklenerek teknik analiz amacıyla kullanılabilecek özel göstergeler oluşturma kabiliyetidir.

Tüm göstergeler, grafiklendirme ismiyle anılan çeşitli çizim stillerinden türetilebilirler. Grafiklendirme (çizim), gösterge tarafından hesaplanan, saklanan ve istek halinde sunulan verilerin görüntülenme şeklini ifade eder. Bu şekilde yedi adet temel çizim tipi bulunmaktadır:

1. Çizgi
2. Kısım (segment)
3. Üst histogram
4. Ok (symbol)
5. Boyalı alan (doldurulmuş kanal)
6. Çubuklar
7. Japon mumları

Her bir çizim tipi için, gösterge değerlerinin depolanacağı bir ile beş adet arasında double tipli dizi gerekir. Bu diziler, uygunluk için gösterge tamponlarıyla ilişkilendirilir. Bir göstergede kullanılacak tamponların sayısı, derleyici direktifleri ile önceden bildirilmelidir, örneğin:

```
#property indicator_buffers 3 // Gösterge tamponlarının sayısı
#property indicator_plots   2 // grafik sayısı
```

Göstergedeki tamponların sayısı, her zaman grafik sayısından fazla veya grafik sayısına eşittir.

Her çizim tipi, farklı renklere ve kurulum ayarlarına sahip olabileceğinden, MQL5'teki çizim tiplerinin gerçek sayısı 18 olarak düşünülebilir:

Çizim	Açıklama	Değer tamponları	Renk tamponları
DRAW_NONE	Çizelge üzerinde görüntülenmez ama karşılık gelen tamponun değerleri Veri Penceresinde görülebilir	1	-
DRAW_LINE	Karşılık gelen tamponu	1	-

Çizim	Açıklama	Değer tamponları	Renk tamponları
	n değerleriyle bir çizgi çizilir (boş değerler istenmez)		
DRAW_SECTION	Karşılık gelen tamponun değerleri arasında çizgi kısımları şeklinde çizilir (genellikle çok fazla boş değeri olur)	1	-
DRAW_HISTOGRAM	Karşılık gelen tamponun değerlerine göre sıfır çizgisinden başlayan bir histogram şeklinde çizilir (boş değerler olabilir)	1	-
DRAW_HISTOGRAM2	İki gösterge tamponunun temelind	2	-

Çizim	Açıklama	Değer tamponları	Renk tamponları
	eki bir histogram şeklinde çizilir (boş değerler olabilir)		
DRAW_ARROW	Sembollerle çizilir (boş değerlere sahip olabilir)	1	-
DRAW_ZIGZAG	DRAW_SECTION stiline benzer ama çubuk üzerine dikey kısımlar da çizebilir	2	-
DRAW_FILLING	İki çizgi arasını renk ile doldurur. Çizelge üzerinde görüntülenmez ama karşılık gelen tamponun değerleri Veri Penceresinde görülebilir	2	-
DRAW_BARS	Çubuklar şeklinde çizilir.	4	-

Çizim	Açıklama	Değer tamponları	Renk tamponları
	Çizelge üzerinde görüntülenmez ama karşılık gelen tamponun değerleri Veri Penceresinde görülebilir		
DRAW_CANDLES	Japon mumları şeklinde çizilir. Çizelge üzerinde görüntülenmez ama karşılık gelen tamponun değerleri Veri Penceresinde görülebilir	4	-
DRAW_COLOR_LINE	Rengini farklı çubuklar üzerinde veya istediğiniz bir anda değiştirebileceğiniz bir çizgi	1	1

Çizim	Açıklama	Değer tamponları	Renk tamponları
DRAW_COLOR_SECTION	DRAW_SECTION stiline benzer ama her kısmın rengi ayrı olarak ayarlanabilir; renk, aynı zamanda dinamik olarak da ayarlanabilir	1	1
DRAW_COLOR_HISTOGRAM	DRAW_HISTOGRAM stiline benzer ama histogramdaki her çubuk farklı renkte olabilir ve renk dinamik olarak ayarlanabilir	1	1
DRAW_COLOR_HISTOGRAM2	DRAW_HISTOGRAM2 stiline benzer ama histogramdaki her çubuk farklı renkte olabilir	2	1

Çizim	Açıklama	Değer tamponları	Renk tamponları
	ve renk dinamik olarak ayarlana bilir		
DRAW_COLOR_ARROW	DRAW_ARROW stiline benzer ama her sembol farklı renkte olabilir. Renk dinamik olarak değiştiril ebilir	1	1
DRAW_COLOR_ZIGZAG	DRAW_ZIGZAG stili gibidir ama kısımları n her biri tekil olarak renklendi rilebilir ve dinamik renk değişimi yapılabili r	2	1
DRAW_COLOR_BARS	DRAW_BARS stili gibidir ama çubukları n her biri tekil olarak renklendi rilebilir ve	4	1

Çizim	Açıklama	Değer tamponları	Renk tamponları
	dinamik renk değişimi yapılabilir		
DRAW_COLOR_CANDLES	DRAW_CANDLES stili gibidir ama mumların her biri tekil olarak renklendirilebilir ve dinamik renk değişimi yapılabilir	4	1

Bir gösterge tamponu ile bir dizi arasındaki fark

Her göstergenin global düzeyinde, bir veya daha fazla double tipi dizi bildirilmelidir; bu diziler daha sonra SetIndexBuffer() fonksiyonu ile bir gösterge tamponu şeklinde kullanılabilir. Gösterge grafiklerinin çiziminde, sadece gösterge tamponlarının değerleri kullanılır, başka herhangi bir dizinin bu amaçla kullanılması mümkün değildir. Gösterge tamponlarının değerleri Veri Penceresinde de gösterilir.

Gösterge tamponu dinamik olmalıdır ve büyüklüğünün belirtilmesi gerekmez - gösterge tamponu olarak kullanılan dizinin büyüklüğü, terminal idari alt-sistemi tarafından otomatik olarak ayarlanır.

Dizinin tampona bağlanmasıyla, indisleme yönü varsayılan olarak, sıradan dizilerdeki gibi ayarlanır ama ArraySetAsSeries() fonksiyonunu kullanarak dizi elemanlarının indisleme yönünü değiştirebilirsiniz. Varsayılan olarak gösterge tamponu, çizim için kullanılacak veriyi (INDICATOR_DATA) saklamak için kullanılır.

Eğer gösterge değerlerinin hesaplanması sırasında, ara hesapların ve çubuklarla ilgili ek verilerin tutulması gerekiyorsa, bunun için kullanılacak dizi bir hesaplama tamponu (INDICATOR_CALCULATIONS) şeklinde bildirilebilir. Ara değerler için sıradan diziler de kullanılabilir ama bu durumda, programcı dizi büyüklüğüyle de ilgilenmek durumundadır.

Bazı grafikler, her bir çubuk için ayrı bir renk ayarlamaya izin verir. Renk hakkındaki bilgileri depolamak için renk tamponları (INDICATOR_COLOR_INDEX) kullanılır. Renk, color tipi bir tamsayıdır,

diğer taraftan, tüm gösterge tamponları double tipinde olmalıdır. Renk değerleri ve yardımcı (INDICATOR_CALCULATIONS) tamponların değerleri, CopyBuffer() fonksiyonu kullanılarak alınamaz.

Gösterge tamponlarının sayısı, #property indicator_buffers number_of_buffers gösterge direktifi ile belirtilmelidir

```
#property indicator_buffers 3  //  göstergenin 3 tamponu var
```

Bir göstergedeki tamponların izin verilen maksimum sayısı 512'dir.

Gösterge Tamponları ile Çizim Arasındaki İlişki

Her grafiklendirme işlemi, bir veya daha fazla gösterge tamponunu temel alır. Yani basit bir mum şeklini görüntülemek için, dört değer gerekir - Open (açılış), High (yüksek), Low (düşük) ve Close (kapanış) fiyatları. Yani, bir göstergeyi mumlarla görüntülemek için, 4 gösterge tamponu ve onlara bağlanacak 4 dizi bildirilmelidir. Örneğin:

```
//--- Dört tamponu olan bir gösterge
#property indicator_buffers 4
//--- Göstergede tek bir çizim var
#property indicator_plots   1
//--- 1 numaralı grafiksel çizim, mumlar şeklinde görüntülenecek
#property indicator_type1   DRAW_CANDLES
//--- Mumlar clrDodgerBlue ile renklendirilecek
#property indicator_color1  clrDodgerBlue
//--- Gösterge tamponları için 4 adet dizi
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

Grafiksel çizimlerde gösterge tamponları, grafik numarasına göre otomatik olarak kullanılır. Grafiklerin numaralandırılması 1 ile başlarken, tamponların numaralandırılması 0 ile başlar. Eğer ilk grafik, 4 gösterge tamponu gerektiriyorsa, çizimde ilk 4 gösterge tamponu kullanılır. Bu dört tampon, SetIndexBuffer() fonksiyonu kullanılarak düzgünce indislenmiş uygun dizilere bağlanmalıdır.

```
//--- Dizilerin tamponlara bağlanması
   SetIndexBuffer(0,OBuffer,INDICATOR_DATA);  // Ilk tampon sıfır indisine karşılık ge
   SetIndexBuffer(1,HBuffer,INDICATOR_DATA);  // Ikinci tampon 1 indisine karşılık gel
   SetIndexBuffer(2,LBuffer,INDICATOR_DATA);  // Üçüncü tampon 2 indisine karşılık gel
   SetIndexBuffer(3,CBuffer,INDICATOR_DATA);  // Dördüncü tampon 3 indisine karşılık ç
```

Mumların çizimi sırasında göstergede sadece ilk 4 tampon kullanacaktır, çünkü "mumların" çizimi ilk numara altında bildirilmiştir.

Örneği değiştirelim ve basit bir çizginin grafiğini ekleyelim - DRAW_LINE. Şimdi çizginin 1 ile, mumların ise 2 ile numaralandırıldığını düşünelim. Tamponların ve grafiklerin sayısı artar.

```
//--- Gösterge 5 tampona sahiptir
```

```
#property indicator_buffers 5
//--- Gösterge 2 grafiğe sahiptir
#property indicator_plots   2
//--- Grafik 1 bir çizgidir
#property indicator_type1   DRAW_LINE
//--- Çizgi rengi clrDodgerRed
#property indicator_color1  clrDodgerRed
//--- Grafik 2 Japon Mumu şeklinde çizilir
#property indicator_type2   DRAW_CANDLES
//--- Mumların rengi clrDodgerBlue
#property indicator_color2  clrDodgerBlue
//--- Gösterge tamponları için 5 dizi
double LineBuffer[];
double OBuffer[];
double HBuffer[];
double LBuffer[];
double CBuffer[];
```

Grafiklerin sırası değişti, şimdi çizgi ilk sırada ve onu Japon mumları takip ediyor. Yani tamponların sırası düzgün - ilk olarak çizgi için sıfır indisi ile bir tampon, ardından mumlar için 4 tampon bildirdik

```
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);  // İlk tampon 0 indisine karşılık ge
//--- Mumlar için, dizilerin tamponlarla bağlanması
   SetIndexBuffer(1,OBuffer,INDICATOR_DATA);     // İkinci tampon 1 indisine karşılık
   SetIndexBuffer(2,HBuffer,INDICATOR_DATA);     // Üçüncü tampon 2 indisine karşılık
   SetIndexBuffer(3,LBuffer,INDICATOR_DATA);     // Dördüncü tampon 3 indisine karşılı
   SetIndexBuffer(4,CBuffer,INDICATOR_DATA);     // Beşinci tampon 4 indisine karşılı
```

Tamponların ve grafiklerin sayısı sadece derleyici direktifleri ile ayarlanabilir, bu özelliklerin, fonksiyonlar kullanılarak dinamik olarak değiştirilmesi mümkün değildir.

Stillerin Renkli Versiyonları

Tablodan da görülebileceği gibi, stiller iki grupta toplanır. İlk grup, isminde **COLOR** kelimesi geçmeyen stilleri içerir; biz bu grubu, temel stiller şeklinde isimlendireceğiz:

- DRAW_LINE
- DRAW_SECTION
- DRAW_HISTOGRAM
- DRAW_HISTOGRAM2
- DRAW_ARROW
- DRAW_ZIGZAG
- DRAW_FILLING
- DRAW_BARS
- DRAW_CANDLES

İkinci grupta, isminde **COLOR** kelimesi geçen stiller yer alır, bu gruba da renkli versiyonlar diyeceğiz:

- DRAW_COLOR_LINE
- DRAW_COLOR_SECTION
- DRAW_COLOR_HISTOGRAM
- DRAW_COLOR_HISTOGRAM2
- DRAW_COLOR_ARROW
- DRAW_COLOR_ZIGZAG
- DRAW_COLOR_BARS
- DRAW_COLOR_CANDLES

Stillerin tüm renkli versiyonları, çizimin her kısmı için ayrı renk belirlenmesine olanak tanıması açısından, temel stillerden ayrılır. Çizimin en küçük parçası bir çubuktur, dolayısıyla renkli stillerin, her bir çubuğun ayrı olarak renklendirilmesine olanak sağladığını söyleyebiliriz.

DRAW_NONE ve DRAW_FILLING stilleri, renkli versiyonları bulunmayan istisnalardır.

Her bir çubuğun rengini ayarlamak için, renk indisini depolayan fazladan bir dizi, renkli versiyona eklenmiştir. Bu indisler, özel bir dizide yer alan bir rengin numarasını gösterir, bahsedilen bu dizi önceden ayarlanmış renkleri içermektedir. Bu renk dizisinin büyüklüğü 64'tür. Yani bir stilin herhangi bir renkli versiyonu, bir grafiğin 64 ayrı renk ile boyanabilmesini sağlar.

Özel bir renk dizisinde kullanılacak renklerin sayısı, gerekli renkleri virgülle ayrılarak girebileceğiniz derleyici direktifi #property indicator_color derleyici direktifi ile ayarlanabilir. Örneğin, bir gösterge içindeki böyle bir giriş:

```
//--- Mumları renklendirmek için 8 renk tanımla (özel dizide depolanırlar)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrL
```

Bu, çizim 1 için özel bir diziye yerleştirilecek 8 renk ayarlandığını belirtmektedir. Bundan sonra program içinde çizim rengini belirtmemiz gerekmez, bunun yerine rengin indisini kullanacağız. Eğer K numaralı çubuğa kırmızı rengi ayarlamak istiyorsak, dizideki indis değerinin, göstergenin renk tamponunda ayarlanmış olması gerekir Kırmızı renk ilk olarak direktifte belirtilir, 0 indis numarasına karşılık gelir.

```
  //--- mum rengini clrRed (kırmızı) olarak ayarla
  col_buffer[buffer_index]=0;
```

Renk kümesi bir defaya mahsus olarak verilmez, PlotIndexSetInteger() fonksiyonu kullanılarak dinamik olarak değiştirilebilir. Örnek:

```
      //--- Her bir indis için, rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
      PlotIndexSetInteger(0,                    //  Bir grafiksel stilin numarası
                          PLOT_LINE_COLOR,      //  Özellik tanımlayıcı
                          plot_color_ind,       //  Rengin girildiği indis
                          color_array[i]);      //  Yeni bir renk
```

Gösterge ve çizim özellikleri

Gösterge grafikleri için özellikler, derleyici direktifleri ve fonksiyonlar gibi araçlar kullanılarak ayarlanabilir. Bununla ilgili daha fazla bilgiyi Gösterge özellikleri ve fonksiyonlar bölümünde okuyabilirsiniz. Gösterge özelliklerinin, özel fonksiyonlar yardımıyla dinamik olarak değiştirilebilmesi, daha esnek özel göstergelerin oluşturulmasını sağlar.

Göstergenin Çizelge Üzerindeki Çizimine Başlanması

Çoğu durumda, algoritmanın koşullarına bağlı olarak, yeni çubukla birlikte göstergenin hesaplanmasına hemen başlamak imkansızdır; belli bir minimum sayıdaki çubuğun geçmişte yer alması gerekir. Örneğin, birçok düzleştirme türünde, N sayıda geçmiş çubuk üzerinden elde edilen bir fiyat dizisi kullanılır ve elde edilen değerlerle göstergenin değerleri hesaplanır.

Bu gibi durumlarda, ya göstergeyi ilk N çubuk için hesaplamanın bir yolu yoktur, ya da bu değerlerin çizelgede görüntülenmesi istenmiyordur ve daha sonraki değerleri hesaplamak için yardımcı durumdadırlar. Göstergenin, geçmişteki ilk N çubuk için çizilmesi istenmiyorsa, N değerini, karşılık gelen grafik için PLOT_DRAW_BEGIN özelliğine ayarlayın:

```
//--- Mumlar için, dizilerin tamponlarla bağlanması
PlotIndexSetInteger(number_of_plot,PLOT_DRAW_BEGIN,N);
```

Burada:

- number_of_plot - sıfırdan indicator_plots-1'e kadar bir değerdir (grafiklerin indislenmesi sıfırdan başlar).
- N - çizelgede, üzerinde göstergenin görüntülenmemesi gereken, geçmişteki ilk çubukların sayısı.

DRAW_NONE

DRAW_NONE stili, bir gösterge tamponunun değerlerinin, çizelgede çizilmeyecek olmasına rağmen Veri Penceresinde görüntülenmek istenmesi durumu için dizayn edilmiştir. Kesinlik değerini artırmak için IndicatorSetInteger(INDICATOR_DIGITS,num_chars) ifadesini OnInit() fonksiyonunda kullanın:

```
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- Veri Penceresinde görüntülenecek değerlerin kesinliğini ayarla
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
```

DRAW_NONE stilinin çizimi için gereken tampon sayısı 1'dir.

Fare ile üzerine gelinen çubuğun numarasını Veri Penceresinde gösteren göstergeye bir örnek. İndisleme yönünün zaman-serilerine karşılık gelmesi, mevcut çubuğun sıfır indisine, en eski çubuğun ise en büyük indise sahip olduğu anlamına gelir.

#1 numaralı grafik için kırmızı renk ayarlanmasına karşın, göstergenin çizelgede hiçbir şey çizmeyeceğini not edin.

```
//+------------------------------------------------------------------+
//|                                                   DRAW_NONE.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             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
//--- gösterge tamponları
double         InvisibleBuffer[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Gösterge tamponunun dizi ile bağlanması
   SetIndexBuffer(0,InvisibleBuffer,INDICATOR_DATA);
//--- Veri Penceresinde görüntülenecek değerlerin kesinliğini ayarla
   IndicatorSetInteger(INDICATOR_DIGITS,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- Eğer bu göstergenin ilk hesaplanışı ise
   if(prev_calculated==0)
     {
      //--- Çubukları ilk sefer için yeniden numarala
      CalcValues(rates_total,close);
      //--- Mevcut çubuğun lastbar içindeki açılış zamanını hatırla
      lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
     }
```

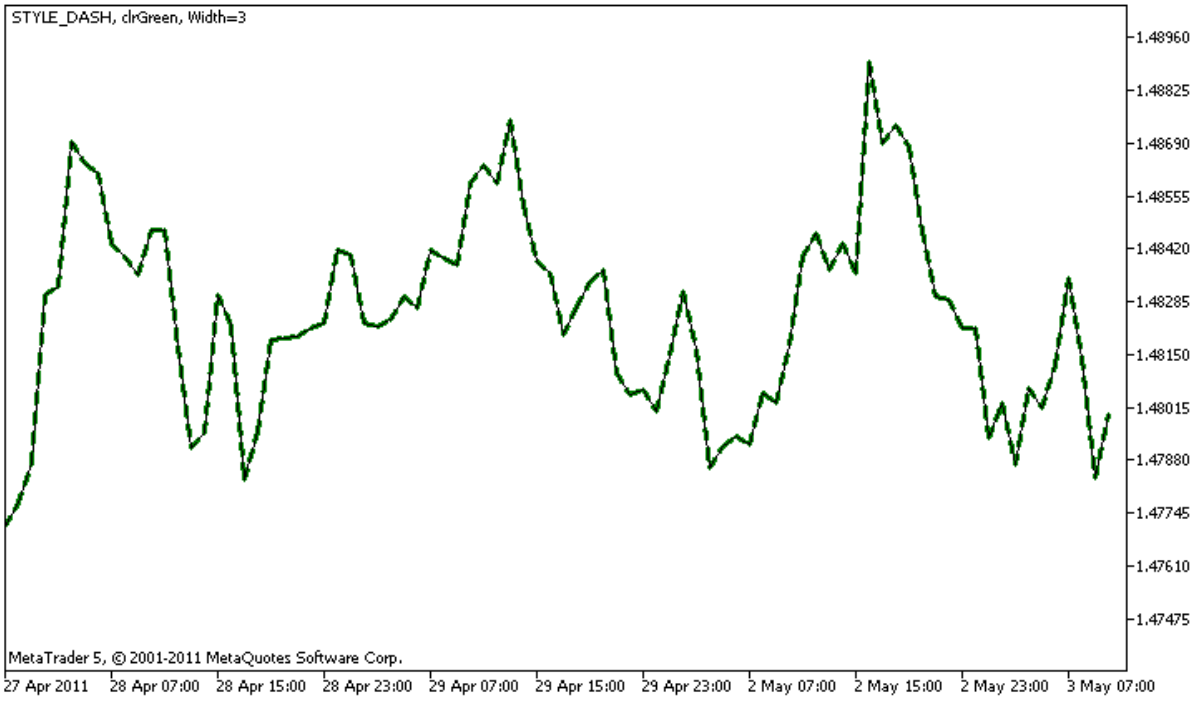
```
   else
     {
      //--- Eğer yeni bir çubuk oluşmuşsa, bunun açılış zamanı lastbar'dan farklı olac
      if(lastbar!=SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE))
        {
         //--- Çubukları bir kez daha numarala
         CalcValues(rates_total,close);
         //--- Mevcut çubuğun lastbar içindeki açılış zamanını güncelle
         lastbar=(datetime)SeriesInfoInteger(_Symbol,_Period,SERIES_LASTBAR_DATE);
        }
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Çubukları zaman serilerindeki gibi indisle                       |
//+------------------------------------------------------------------+
void CalcValues(int total,double const  &array[])
  {
//--- Gösterge tamponunun indisleme yönünü zaman-serilerindeki gibi yap
   ArraySetAsSeries(InvisibleBuffer,true);
//--- Her çubuğu kendi numarasıyla doldur
   for(int i=0;i<total;i++) InvisibleBuffer[i]=i;
  }
```

DRAW_LINE

DRAW_LINE stili, gösterge tamponunun değerlerine göre, belirtilen renkte bir çizgi çizer. Çizgilerin genişlikleri, stilleri ve renkleri, derleyici direktifleri ile veya dinamik olarak PlotIndexSetInteger() fonksiyonunu kullanarak ayarlanabilir. Çizim özelliklerinin dinamik olarak değiştirilmesi, göstergelerin "canlılaştırılmasını" sağlar, bu şekilde göstergelerin görünümleri mevcut duruma göre değişir.

DRAW_LINE stilinin çizimi için gereken tampon sayısı 1'dir.

Çubukların Close (kapanış) fiyatlarını kullanarak çizgi çizen bir gösterge örneği. Çizgi rengi, kalınlığı ve stili her N=5 tikte bir rassal olarak değişir.



DRAW_LINE stiline sahip **plot1** için, özelliklerin başlangıçta #property derleyici direktifi kullanılarak belirlendiğini; sonrasında ise üç özelliğin, OnCalculate() fonksiyonu içinde önceden hazırlanmış bir listeden rassal olarak ayarlandığını lütfen not ediniz. N parametresi, el ile yapılandırma olasılığı göz önüne alınarak, göstergenin dışsal parametreleri içinde ayarlanır (göstergenin özellikler penceresindeki Veriler sekmesi).

```
//+------------------------------------------------------------------+
//|                                                    DRAW_LINE.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_LINE stilini betimleyen bir gösterge"
#property description "Close (kapanış) değerleri üzerinde, belirtilen renkte bir çizgi
#property description "Çizgilerin renk, genişlik ve stilleri, her N tikten sonra"
#property description "rassal olarak değiştirilir"
```

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- Çizgi özellikleri, derleyici direktifleri ile ayarlandı
#property indicator_label1  "Line"      // Veri Penceresi için gösterge ismi
#property indicator_type1   DRAW_LINE   // Çizim stili çizgi olarak belirlendi
#property indicator_color1  clrRed      // Çizgi rengi
#property indicator_style1  STYLE_SOLID // Çizgi stili
#property indicator_width1  1           // Çizgi genişliği
//--- giriş parametreleri
input int      N=5;         // Değiştirilecek tik sayısı
//--- Grafik için bir gösterge tamponu
double         LineBuffer[];
//--- Renkleri depolayacak bir dizi
color colors[]={clrRed,clrBlue,clrGreen};
//--- Çizgi stillerinin depolanması için bir dizi
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Gösterge tamponunun dizi ile bağlanması
   SetIndexBuffer(0,LineBuffer,INDICATOR_DATA);
//--- Pseudo-rassal sayı oluşturucusunun başlatılması
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- Çizginin stilini,rengini ve genişliğini değiştirmek için tikleri hesapla
   ticks++;
//--- Tikler önemli bir sayıda biriktirilmişse
   if(ticks>=N)
```

```
      {
       //--- Çizgi özelliklerini değiştir
       ChangeLineAppearance();
       //--- Tik sayacını sıfırla
       ticks=0;
      }

//--- Gösterge değerlerinin hesaplanması için bir blok
   for(int i=0;i<rates_total;i++)
     {
      LineBuffer[i]=close[i];
     }

//--- Fonksiyonun sonraki çağrıları için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Çizilen gösterge çizgisinin görünümünü değiştirir                |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Çizgi özelliklerine dair bilginin biçimlendirilmesi için bir dizgi
   string comm="";
//--- Çizgi rengini değiştirmek için bir blok
//--- Rassal bir değer al
   int number=MathRand();
//--- Bölen, colors[] dizisinin boyutuna eşit
   int size=ArraySize(colors);
//--- Bölümden kalan tam-sayı değeri şeklinde yeni bir renk seçmek için, indis değeri
   int color_index=number%size;
//--- Rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Çizgi rengini ayarla
   comm=comm+(string)colors[color_index];

//--- Çizgi kalınlığını değiştirmek için bir blok
   number=MathRand();
//--- Tam-sayı bölümden kalan genişliği al
   int width=number%5; // Genişlik 0'dan 4'e ayarlanır
//--- Rengi PLOT_LINE_WIDTH özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Çizgi kalınlığını yaz
   comm=comm+", Width="+IntegerToString(width);

//--- Çizgi stilini değiştirmek için bir blok
   number=MathRand();
//--- Bölen, syles dizisinin büyüklüğüne eşit
   size=ArraySize(styles);
//--- Yeni bir stil seçmek için, indis değerini, bölümden kalan tam-sayı değeri şeklin
```

```
   int style_index=number%size;
//--- Rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Çizgi stilini yaz
   comm=EnumToString(styles[style_index])+", "+comm;
//--- Bir yorum kullanarak, çizelge üzerinde bilgi göster
   Comment(comm);
  }
```

DRAW_SECTION

DRAW_SECTION stili, gösterge tamponunun değerlerine göre, kısımlar (parçalar) çizer. Çizgilerin kalınlık, renk ve stilleri, DRAW_LINE stiline benzer şekilde, (derleyici direktifleri ile veya dinamik olarak PlotIndexSetInteger() fonksiyonunun kullanımıyla) belirtilebilir. Çizim özelliklerinin dinamik olarak değiştirilmesi, göstergelerin "canlılaştırılmasını" sağlar, bu şekilde göstergelerin görünümleri mevcut duruma göre değişir.

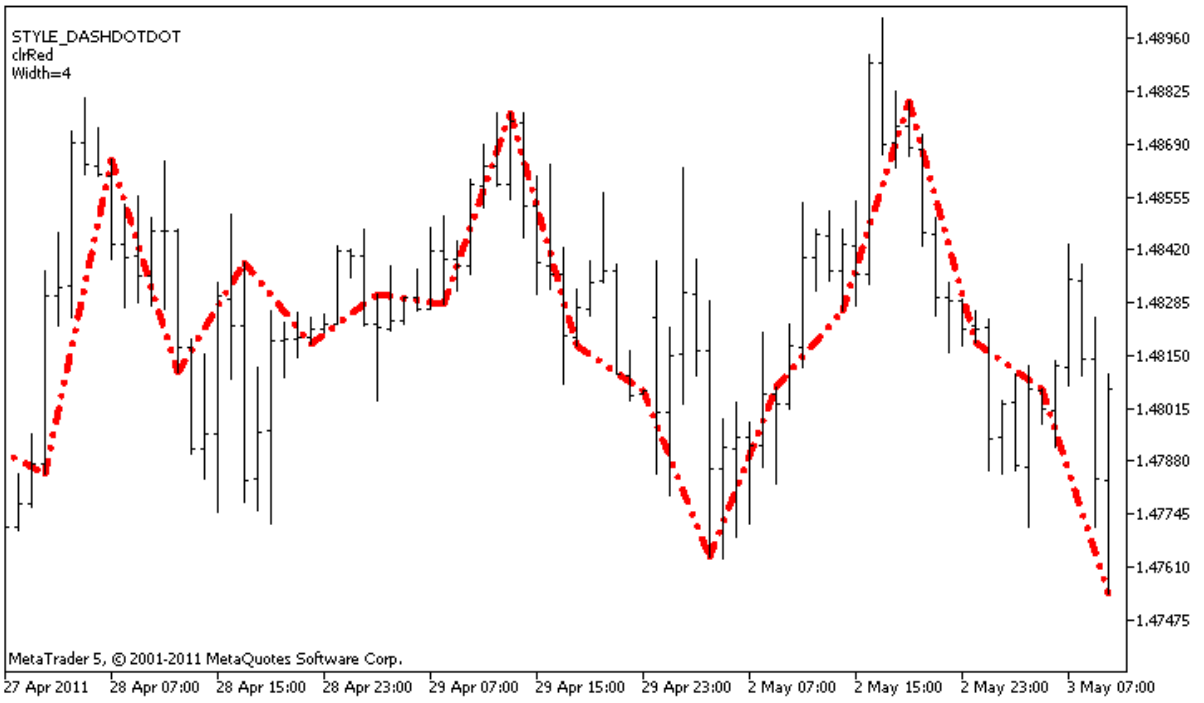
Parçalar (kısımlar), gösterge tamponunun boş olmayan bir değerinden, yine boş olmayan bir değerine kadar çizilir ve boş değerler gözardı edilir. Bir değerin "boş" olarak göz önüne alınacağını belirtmek için, bu değeri PLOT_EMPTY_VALUE özelliği ile ayarlayın: Örneğin, gösterge, sıfır olmayan değerler üzerindeki parçalardan oluşan bir dizi şeklinde çizilecekse, sıfır değerini boş değer olarak ayarlamalısınız:

```
//--- 0 (boş) değer, çizime katılmayacak
   PlotIndexSetDouble(çizim_indisi_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```

Gösterge tamponlarının değerlerini her zaman açıkça doldurun, atlamak istediğiniz çubukları boş değer olarak ayarlayın.

DRAW_SECTION stilinin çizilmesi için gereken tampon sayısı 1'dir.

High ve Low fiyatların arasında parçalar çizen gösterge örneği. **Tüm** parçaların renkleri, genişlikleri ve stilleri, her **N** tikte bir değişir.



DRAW_SECTION stiline sahip **plot1** için, özelliklerin başlangıçta #property derleyici direktifi kullanılarak belirlendiğini; sonrasında ise üç özelliğin, OnCalculate() fonksiyonu içinde önceden hazırlanmış bir listeden rassal olarak ayarlandığını lütfen not ediniz. N parametresi, el ile yapılandırma olasılığı göz önüne alınarak, göstergenin dışsal parametreleri içinde ayarlanır (göstergenin özellikler penceresindeki Veriler sekmesi).

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

```
//|                                                  DRAW_SECTION.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_SECTION stilini betimleyen bir gösterge"
#property description "Her bars çubuk üzerine düz parçalar çizer"
#property description "Parçaların renk, genişlik ve stilleri, her N tikten sonra"
#property description "rassal olarak değiştirilir"

#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
//--- giriş parametreleri
input int      bars=5;           // Parçaların çubuk bazında uzunluğu
input int      N=5;              // Parçaların stillerinin değiştirileceği tik sayısı
//--- Grafik için bir gösterge tamponu
double         SectionBuffer[];
//--- Parçaların sonlarını hesaplamak için ek bir değişken
int            divider;
//--- Renkleri depolayacak bir dizi
color colors[]={clrRed,clrBlue,clrGreen};
//--- Çizgi stillerinin depolanması için bir dizi
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Gösterge tamponunun dizi ile bağlanması
   SetIndexBuffer(0,SectionBuffer,INDICATOR_DATA);
//--- 0 (boş) değer, çizime katılmayacak
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- Gösterge parametresini kontrol et
   if(bars<=0)
     {
      PrintFormat("Hatalı parametre değeri bar=%d",bars);
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=2*bars;
```

```
//---+
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- Çizginin stilini,rengini ve genişliğini değiştirmek için tikleri hesapla
   ticks++;
//--- Tikler önemli bir sayıda biriktirilmişse
   if(ticks>=N)
     {
      //--- Çizgi özelliklerini değiştir
      ChangeLineAppearance();
      //--- Tik sayacını sıfırla
      ticks=0;
     }

//--- Gösterge değerlerinin hesaplanmaya başlanacağı çubuğun numarası
   int start=0;
//--- Gösterge daha önce hesaplanmışsa, başlangıcı bir önceki çubuğa al
   if(prev_calculated>0) start=prev_calculated-1;
//--- Burada göstergenin tüm değerleri hesaplanır
   for(int i=start;i<rates_total;i++)
     {
      //--- çubuk sayısının 2*bars değerine bölümünden kalanı al
      int rest=i%divider;
      //--- Eğer sayı 2*bars değerine bölünebiliyorsa
      if(rest==0)
        {
         //--- Parçanın sonunu bu çubuğun High (yüksek) fiyatına ayarla
         SectionBuffer[i]=high[i];
        }
      //---Bölümden kalan 'bars' değerine eşitse,
      else
        {
         //--- Parçanın sonunu bu çubuğun High (yüksek) fiyatına ayarla
         if(rest==bars) SectionBuffer[i]=low[i];
```

```
         //--- Hiçbir değişim olmadıysa çubuğu gözardı et - 0 olarak ayarla
         else SectionBuffer[i]=0;
        }
     }
//--- Fonksiyonun sonraki çağrıları için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Göstergedeki parçaların görünümünü değiştirir                   |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Çizgi özelliklerine dair bilginin biçimlendirilmesi için bir dizgi
   string comm="";
//--- Çizgi renginin değişimi için bir blok
   int number=MathRand(); // Rassal bir sayı al
//--- Bölen, colors[] dizisinin boyutuna eşit
   int size=ArraySize(colors);
//--- Bölümden kalan tam-sayı değeri şeklinde yeni bir renk seçmek için, indis değeri
   int color_index=number%size;
//--- Rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Çizgi rengini ayarla
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Çizgi kalınlığını değiştirmek için bir blok
   number=MathRand();
//--- Tam-sayı bölümden kalan genişliği al
   int width=number%5;   // Genişlik 0'dan 4'e ayarlandı
//--- Kalınlığı ayarla
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Çizgi kalınlığını yaz
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Çizgi stilini değiştirmek için bir blok
   number=MathRand();
//--- Bölen, syles dizisinin büyüklüğüne eşit
   size=ArraySize(styles);
//--- Yeni bir stil seçmek için, indis değerini, bölümden kalan tam-sayı değeri şeklin
   int style_index=number%size;
//--- Çizgi stilini ayarla
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Çizgi stilini yaz
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- Bir yorum kullanarak, çizelge üzerinde bilgi göster
   Comment(comm);
  }
```

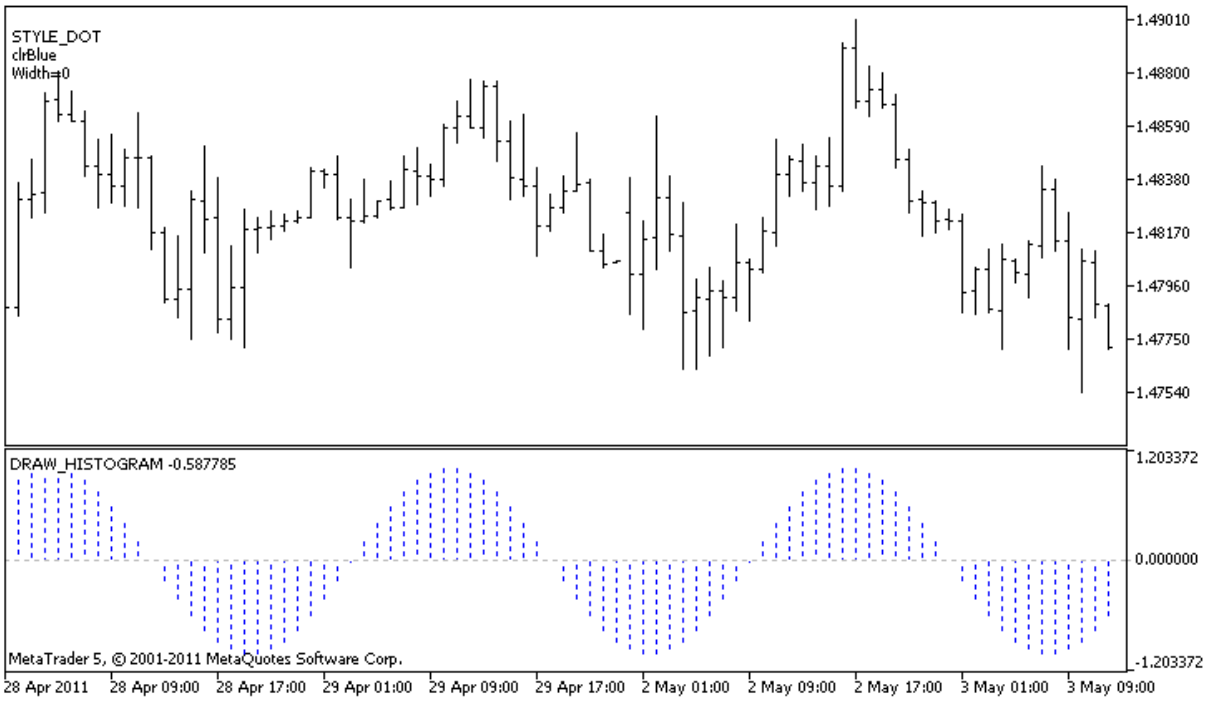
DRAW_HISTOGRAM

DRAW_HISTOGRAM stili, sıfır seviyesinden belirtilen değere kadar, belirtilen renkte bir sütun dizisi şeklinde bir histogram çizer. Değerler gösterge tamponundan alınır. Sütunların kalınlık, renk ve stilleri, DRAW_LINE stiline benzer şekilde (derleyici direktifleri ile veya dinamik olarak PlotIndexSetInteger() fonksiyonunun kullanımıyla) belirtilebilir. Çizim özelliklerinin dinamik olarak değişmesi, mevcut duruma bağlı olarak, histogram görünümünün de değişmesini sağlar.

Her çubuk için sıfır seviyesinden bir sütun çizildiğinden, DRAW_HISTOGRAM stilinin ayrı bir pencerede kullanılması daha iyi olacaktır. Bu çizim tipi genellikle osilatör tipi göstergeler çizmek için kullanılır, örneğin, Bears Power veya OsMA. Görüntülenmeyecek boş değerler için sıfır değeri belirtilmelidir.

DRAW_HISTOGRAM stilinin çizimi için gereken tampon sayısı 1'dir.

MathSin() fonksiyonu temelinde, belirtilen bir renk ile bir sinüsoid çizen gösterge örneği. **Tüm** histogram sütunlarının renkleri, genişlikleri ve stilleri, her **N** tikte bir rassal olarak değişir. "bars" parametresi, sinüsoidin periyodunu belirtir, yani belirtilen sayıdaki çubuğun ardından sinüsoid döngüyü tekrar edecektir.



DRAW_HISTOGRAM stilindeki **plot1** için, özelliklerin başlangıçta #property derleyici direktifi kullanılarak belirlendiğini; sonrasında ise üç özelliğin, OnCalculate() fonksiyonu içinde önceden hazırlanmış bir listeden rassal olarak ayarlandığını lütfen not ediniz. N parametresi, el ile yapılandırma olasılığı göz önüne alınarak, göstergenin dışsal parametreleri içinde ayarlanır (göstergenin özellikler penceresindeki Veriler sekmesi).

```
//+------------------------------------------------------------------+
//|                                                DRAW_HISTOGRAM.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
```

```
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_HISTOGRAM stilini betimleyen bir gösterge"
#property description "Ayrı bir pencerede, histogram şeklinde bir sinüsoid çizer"
#property description "Çubukların renk ve genişliği, her N tikten sonra"
#property description "rassal olarak değiştirilir"
#property description "bars parametresi, sinüsoid döngüsü içindeki çubuk sayısını ayar

#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
//--- giriş parametreleri
input int      bars=30;          // Çubuk sayısıyla, sinüsoid periyodu
input int      N=5;              // Histogramı değiştirmek için gereken tik sayısı
//--- gösterge tamponları
double         HistogramBuffer[];
//--- Bars parametresi ile çarpıldığında, 2Pi'lik açıyı radyan olarak alabilmemiz için
double    multiplier;
//--- Renkleri depolayacak bir dizi
color colors[]={clrRed,clrBlue,clrGreen};
//--- Çizgi stillerinin depolanması için bir dizi
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,HistogramBuffer,INDICATOR_DATA);
//--- Çarpan değerini hesapla
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("bars=%d değerini 1'den büyük olarak ayarla",bars);
      //--- Göstergenin erken sonlandırılması
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
```

```
//+------------------------------------------------------------------+
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;
//--- Çizginin stilini,rengini ve genişliğini değiştirmek için tikleri hesapla
   ticks++;
//--- Tikler önemli bir sayıda biriktirilmişse
   if(ticks>=N)
     {
      //--- Çizgi özelliklerini değiştir
      ChangeLineAppearance();
      //--- Tik sayacını sıfırla
      ticks=0;
     }

//--- Gösterge değerlerini hesapla
   int start=0;
//--- Daha önceki OnCalculate çağrısında hesaplanmışsa
   if(prev_calculated>0) start=prev_calculated-1; // hesaplama başlangıcını son seferi
//--- Gösterge tamponunu değerlerle doldur
   for(int i=start;i<rates_total;i++)
     {
      HistogramBuffer[i]=sin(i*multiplier);
     }
//--- Fonksiyonun sonraki çağrıları için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Gösterge çizgilerinin görünümünü değiştirir                      |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Çizgi özelliklerine dair bilginin biçimlendirilmesi için bir dizgi
   string comm="";
//--- Çizgi rengini değiştirmek için bir blok
   int number=MathRand(); // Rassal bir sayı al
//--- Bölen, colors[] dizisinin boyutuna eşit
   int size=ArraySize(colors);
//--- Bölümden kalan tam-sayı değeri şeklinde yeni bir renk seçmek için, indis değeri
   int color_index=number%size;
```

```
//--- Rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Çizgi rengini ayarla
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Çizgi kalınlığını değiştirmek için bir blok
   number=MathRand();
//--- Tam-sayı bölümden kalan genişliği al
   int width=number%5;   // Genişlik 0'dan 4'e ayarlandı
//--- Kalınlığı ayarla
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Çizgi kalınlığını yaz
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Çizgi stilini değiştirmek için bir blok
   number=MathRand();
//--- Bölen, syles dizisinin büyüklüğüne eşit
   size=ArraySize(styles);
//--- Yeni bir stil seçmek için, indis değerini, bölümden kalan tam-sayı değeri şeklir
   int style_index=number%size;
//--- Çizgi stilini ayarla
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Çizgi stilini yaz
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- Bir yorum kullanarak, çizelge üzerinde bilgi göster
   Comment(comm);
  }
```

DRAW_HISTOGRAM2

The DRAW_HISTOGRAM2 stili, belirtilen renkte bir histogram çizer - iki gösterge tamponunun değerlerini kullanarak dikey parçalar çizer. Parçaların kalınlıkları, renkleri ve stilleri, DRAW_LINE stiline benzer şekilde, (derleyici direktifleri ile veya dinamik olarak PlotIndexSetInteger() fonksiyonunun kullanımıyla) belirtilebilir. Çizim özelliklerinin dinamik olarak değişmesi, mevcut duruma bağlı olarak, histogram görünümünün de değişmesini sağlar.

DRAW_HISTOGRAM2 stili, ayrı bir pencerede veya çizelge ana penceresinde kullanılabilir. Boş değerler için hiçbir şey çizilmez; gösterge tamponundaki tüm değerler açıkça ayarlanmalıdır. Tamponlar sıfır değeri ile başlatılamazlar.

DRAW_HISTOGRAM2 stilinin çizimi için gereken gösterge tamponu sayısı 2'dir.

Open (açılış) ve Close (kapanış) fiyatları arasında, belirtilen renk ve kalınlıkta bir dikey parça - histogramı - çizen gösterge örneği. **Tüm** histogram sütunlarının renkleri, genişlikleri ve stilleri, her **N** tikte bir değişir. Gösterge başlangıcında, OnInit() fonksiyonu içinde, göstergenin çizilmeyeceği günün numarası - invisible_day - rassal olarak ayarlanır. Bu amaçla boş değer şu şekilde ayarlanır, PLOT_EMPTY_VALUE=0:

```
//--- Bir boş değer ayarla
   PlotIndexSetDouble(çizim_indisi_DRAW_SECTION,PLOT_EMPTY_VALUE,0);
```

DRAW_HISTOGRAM2 stilindeki **plot1** için, özelliklerin başlangıçta #property derleyici direktifi kullanılarak belirlendiğini; sonrasında ise üç özelliğin, OnCalculate() fonksiyonu içinde önceden hazırlanmış bir listeden rassal olarak ayarlandığını lütfen not ediniz. N parametresi, el ile yapılandırma olasılığı göz önüne alınarak, göstergenin dışsal parametreleri içinde ayarlanır (göstergenin özellikler penceresindeki Veriler sekmesi).

```
//+------------------------------------------------------------------+
//|                                                DRAW_HISTOGRAM2.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
```

```
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_HISTOGRAM2 stilini betimleyen bir gösterge"
#property description "Her bir çubuk üzerinde Open ve Close arasındaki bir kısımı çize
#property description "renk, genişlik ve stil, her N tikten sonra"
#property description "rassal olarak değiştirilir"

#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
//--- giriş parametreleri
input int      N=5;              // Histogramı değiştirmek için gereken tik sayısı
//--- gösterge tamponları
double         Histogram_2Buffer1[];
double         Histogram_2Buffer2[];
//--- Haftanın, göstergenin çizilmeyeceği günü
int invisible_day;
//--- Renkleri depolayacak bir dizi
color colors[]={clrRed,clrBlue,clrGreen};
//--- Çizgi stillerinin depolanması için bir dizi
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,Histogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Histogram_2Buffer2,INDICATOR_DATA);
//--- Bir boş değer ayarla
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- 0 ile 5 arası bir rassal sayı al
   invisible_day=MathRand()%6;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
```

```
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;
//--- Çizginin stilini,rengini ve genişliğini değiştirmek için tikleri hesapla
   ticks++;
//--- Tikler önemli bir sayıda biriktirilmişse
   if(ticks>=N)
     {
      //--- Çizgi özelliklerini değiştir
      ChangeLineAppearance();
      //--- Tik sayacını sıfırla
      ticks=0;
     }

//--- Gösterge değerlerini hesapla
   int start=0;
//--- Haftanın günlerini her bir çubuğun açılışını kullanarak almak için
   MqlDateTime dt;
//--- Daha önceki OnCalculate çağrısında hesaplanmışsa
   if(prev_calculated>0) start=prev_calculated-1; // hesaplama başlangıcını son seferi
//--- Gösterge tamponunu değerlerle doldur
   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];
        }
     }
//--- Fonksiyonun sonraki çağrıları için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Gösterge çizgilerinin görünümünü değiştirir                      |
```

```
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Çizgi özelliklerine dair bilginin biçimlendirilmesi için bir dizgi
   string comm="";
//--- Çizgi renginin değişimi için bir blok
   int number=MathRand(); // Rassal bir sayı al
//--- Bölen, colors[] dizisinin boyutuna eşit
   int size=ArraySize(colors);
//--- Bölümden kalan tam-sayı değeri şeklinde yeni bir renk seçmek için, indis değerir
   int color_index=number%size;
//--- Rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Çizgi rengini ayarla
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Çizgi kalınlığını değiştirmek için bir blok
   number=MathRand();
//--- Tam-sayı bölümden kalan genişliği al
   int width=number%5;   // Genişlik 0'dan 4'e ayarlandı
//--- Çizgi genişliğini ayarla
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Çizgi kalınlığını yaz
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Çizgi stilini değiştirmek için bir blok
   number=MathRand();
//--- Bölen, syles dizisinin büyüklüğüne eşit
   size=ArraySize(styles);
//--- Yeni bir stil seçmek için, indis değerini, bölümden kalan tam-sayı değeri şeklir
   int style_index=number%size;
//--- Çizgi stilini ayarla
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Çizgi stilini yaz
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- Hesaplamada atlanan gün hakkında bilgi ekle
   comm="\r\nNot plotted day - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+comm;
//--- Bir yorum kullanarak, çizelge üzerinde bilgi göster
   Comment(comm);
  }
```

DRAW_ARROW

DRAW_ARROW stili, gösterge değerindeki değere bağlı olarak, belirtilen renkte oklar çizer (Wingdings semboller kümesi). Sembollerin kalınlık ve genişlikleri DRAW_LINE stiline benzer şekilde belirtilebilir - derleyici direktifleri ile veya dinamik olarak PlotIndexSetInteger() fonksiyonunun kullanımıyla. Çizim özelliklerinin dinamik olarak değişmesi, mevcut duruma bağlı olarak, gösterge görünümünün değişmesini sağlar.

Sembol kodu, PLOT_ARROW özelliği kullanılarak ayarlanır.

```
//--- PLOT_ARROW'da çizmek için Wingdings yazı tipinden bir sembolün kodunu tanımla
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

PLOT_ARROW için varsayılan değer 159'dur (daire).

Her ok, gerçekte bir yüksekliğe ve tutturma noktasına sahip olan bir semboldür ve çizelge üzerinde bazı önemli bilgileri örtebilir (örneğin bir çubuğun kapanış fiyatı). Bu nedenle, çizelge ölçeğine bağlı olmayan, piksel bazında bir dikey kaydırmayı ayrıyeten belirtebiliriz. Bu durumda oklar, belirtilen piksel sayısı kadar aşağı kaydırılacaktır, ayrıca gösterge değeri de aynı kalacaktır:

```
//--- Oklar için piksel bazında dikey kaydırma ayarla
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

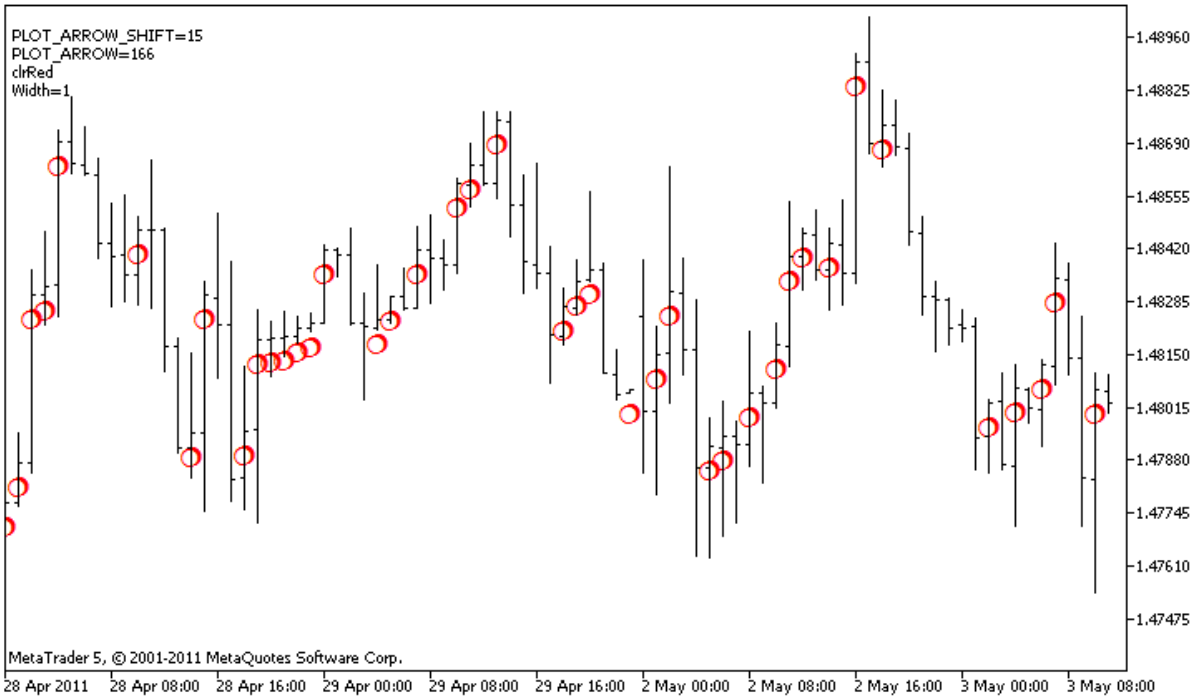
PLOT_ARROW_SHIFT'in negatif değerleri, okun yukarı kaydırılacağı anlamına gelir, pozitif bir değer ise aşağı kaydırma demektir.

DRAW_ARROW stili, bir alt pencerede veya çizelge ana penceresinde kullanılabilir. Boş değerler çizilmez ve "Veri Penceresinde" gösterilmezler; bu yüzden gösterge tamponlarındaki tüm değerler açıkça ayarlanmalıdır. Tamponlar sıfır değeri ile başlatılamazlar.

```
//--- Bir boş değer ayarla
   PlotIndexSetDouble(çizim_indisi_DRAW_ARROW,PLOT_EMPTY_VALUE,0);
```

DRAW_ARROW stilini çizmek için gereken tampon sayısı 1'dir.

Bir önceki çubuğun kapanış fiyatından daha yüksek kapanış fiyatına sahip olan çubuğun üzerine ok çizen gösterge örneği **Tüm** okların renkleri, genişlikleri, kaydırma değerleri ve sembol kodları, her **N** tikte bir rassal olarak değişir.



DRAW_ARROW stilindeki **plot1** örneğinde, renk ve boyut özellikleri, #property derleyici direktifi kullanılarak belirlenir, sonrasında ise OnCalculate() fonksiyonunda, özellikler rassal olarak ayarlanır. N parametresi, el ile yapılandırma olasılığı göz önüne alınarak, göstergenin dışsal parametreleri içinde ayarlanır (göstergenin özellikler penceresindeki Veriler sekmesi).

```
//+------------------------------------------------------------------+
//|                                                   DRAW_ARROW.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_ARROW stilini tanıtmak için bir gösterge"
#property description "Unicode karakteriyle, çizelge üzerinde oklar çizer"
#property description "Okun rengi, genişliği, kaydırma değeri ve sembol kodu, her N ti
#property description "rassal olarak değiştirilir"
#property description "code parametresi, temel değeri ayarlar: code=159 (daire)"

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- grafik Histogram
#property indicator_label1  "Arrows"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrGreen
#property indicator_width1  1
//--- giriş parametreleri
input int      N=5;          // Değiştirilecek tik sayısı
```

```
input ushort   code=159;    // DRAW_ARROW içinde çizilecek sembol kodu
//--- Grafik için bir gösterge tamponu
double         ArrowsBuffer[];
//--- Renkleri depolayacak bir dizi
color colors[]={clrRed,clrBlue,clrGreen};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,ArrowsBuffer,INDICATOR_DATA);
//--- PLOT_ARROW içinde çizilecek sembol kodunu tanımla
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- Oklar için piksel bazında dikey kaydırma ayarla
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- Boş değer 0 olarak ayarla
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- Rengin, boyutun, kaydırma değerinin ve sembol kodunun değiştirileceği tikleri he
   ticks++;
//--- Tikler önemli bir sayıda biriktirilmişse
   if(ticks>=N)
     {
      //--- Çizgi özelliklerini değiştir
      ChangeLineAppearance();
      //--- Tik sayacını sıfırla
      ticks=0;
     }

//--- Gösterge değerlerinin hesaplanması için bir blok
   int start=1;
```

```
   if(prev_calculated>0) start=prev_calculated-1;
//--- Hesap döngüsü
   for(int i=1;i<rates_total;i++)
     {
      //--- mevcut Close (kapanış) fiyatı öncekinden yüksekse, bir ok çiz
      if(close[i]>close[i-1])
         ArrowsBuffer[i]=close[i];
      //--- Aksi durumda sıfır değeri ayarla
      else
         ArrowsBuffer[i]=0;
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Sembollerin göstergedeki görünümünü değiştir                     |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Gösterge özellikleri hakkında alınan bilgiyi biçimlendirmek için bir dize
   string comm="";
//--- Ok rengini değiştirmek için bir blok
   int number=MathRand(); // Rassal bir sayı al
//--- Bölen, colors[] dizisinin boyutuna eşit
   int size=ArraySize(colors);
//--- Bölümden kalan tam-sayı değeri şeklinde yeni bir renk seçmek için, indis değeri
   int color_index=number%size;
//--- Rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Çizgi rengini ayarla
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Okların boyutunu değiştirmek için bir blok
   number=MathRand();
//--- Tam-sayı bölümden kalan genişliği al
   int width=number%5;   // Boyut 0'dan 4'e ayarlandı
//--- Rengi PLOT_LINE_WIDTH özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Ok boyutunu yaz
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Ok kodunu (PLOT_ARROW) değiştirmek için bir blok
   number=MathRand();
//--- Yeni ok kodunu hesaplamak için tam-sayı bölümünden kalanı al (0'dan 19'a)
   int code_add=number%20;
//--- code+code_add işleminin sonucu olarak yeni sembol kodunu ayarla
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- PLOT_ARROW sembol kodunu yaz
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;
```

```
//--- Okların dikey kaydırma değerini piksel bazında değiştirmek için bir blok
   number=MathRand();
//--- Kaydırma değerini tamsayı bölümünün kalanı olarak al
   int shift=20-number%41;
//--- Yeni kaydırma değerini -20'den 20'ye ayarla
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- PLOT_ARROW_SHIFT kaydırma değerini yaz
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;

//--- Bir yorum kullanarak, çizelge üzerinde bilgi göster
   Comment(comm);
  }
```

DRAW_ZIGZAG

DRAW_ZIGZAG stili iki ayrı gösterge tamponunun değerleri temelinde parçalar çizer. Bu stil, DRAW_SECTION stiline benzer ama onun aksine, her iki tampon değerinin de tanımlı olduğu çubukların üzerinde dikey parçalar çizilmesine olanak sağlar. Parçalar ilk gösterge tamponunun değerinden ikinci tamponun değerine çizilir. Tamponların hiçbiri baştan sona boş değerler içeremez; böyle bir durumda hiçbir şey çizilmeyecektir.

Çizgilerin kalınlık, renk ve stilleri, DRAW_SECTION stiline benzer şekilde, derleyici direktifleri ile veya dinamik olarak PlotIndexSetInteger() fonksiyonunun kullanımıyla belirtilebilir. Çizim özelliklerinin dinamik olarak değiştirilmesi, göstergelerin "canlılaştırılmasını" sağlar, bu şekilde göstergelerin görünümleri mevcut duruma göre değişir.

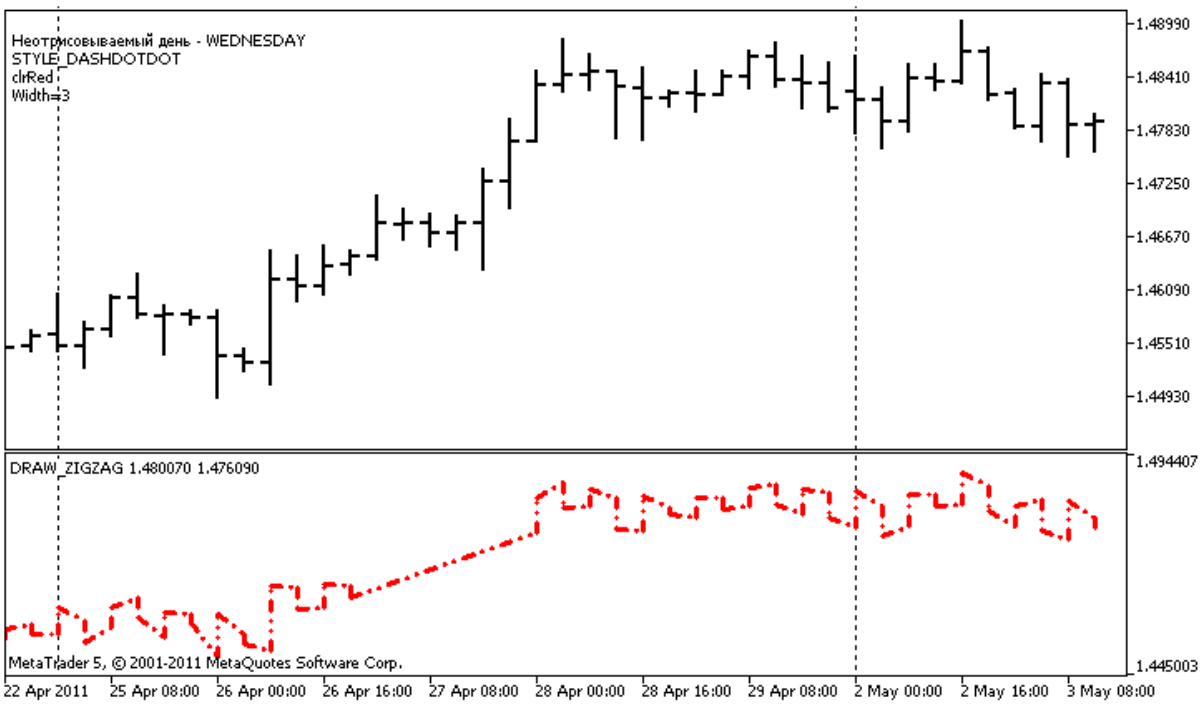
Parçalar, gösterge tamponunun boş olmayan bir değerinden, yine boş olmayan bir değerine kadar çizilir. Bir değerin "boş" olacağını belirtmek için, değeri PLOT_EMPTY_VALUE özelliğiyle ayarlayın:

```
//--- 0 (boş) değer, çizime katılmayacak
   PlotIndexSetDouble(çizim_indisi_DRAW_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

Gösterge tamponlarının değerlerini her zaman açıkça doldurun, atlamak istediğiniz çubukları boş değer olarak ayarlayın.

DRAW_ ZIGZAG stilinin çizimi için gereken tampon sayısı 2'dir.

High ve Low fiyatların temelinde bir testere şekli çizen gösterge örneği. Zigzag çizgilerinin renkleri, kalınlıkları ve stilleri **N** tikte bir rassal olarak değişir.



DRAW_ZIGZAG stiline sahip **plot1** için, özelliklerin başlangıçta #property derleyici direktifi kullanılarak belirlendiğini; sonrasında ise bu özelliklerin, OnCalculate() fonksiyonu içinde önceden hazırlanmış bir listeden rassal olarak ayarlandığını lütfen not ediniz. N parametresi, el ile yapılandırma olasılığı göz önüne alınarak, göstergenin dışsal parametreleri içinde ayarlanır (göstergenin özellikler penceresindeki Veriler sekmesi).

```
//+------------------------------------------------------------------+
//|                                                  DRAW_ZIGZAG.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_ZIGZAG stilini betimleyen bir gösterge"
#property description "düz parçalar halinde, bir günlük çubuk atlayarak bir \"testere\
#property description "Atlanacak gün, gösterge başlangıcında rassal olarak seçilir"
#property description "Parçaların renk, genişlik ve stilleri, her N tikten sonra"
#property description " rassal olarak değiştirilir"

#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
//--- giriş parametreleri
input int      N=5;              // Değişim için gerekli tik sayısı
//--- gösterge tamponları
double         ZigZagBuffer1[];
double         ZigZagBuffer2[];
//--- Haftanın, göstergenin çizilmeyeceği günü
int invisible_day;
//--- Renkleri depolayacak bir dizi
color colors[]={clrRed,clrBlue,clrGreen};
//--- Çizgi stillerinin depolanması için bir dizi
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Dizilerin gösterge tamponlarına bağlanması
   SetIndexBuffer(0,ZigZagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ZigZagBuffer2,INDICATOR_DATA);
//--- 0 ile 6 arasında bir rassal sayı al, bu numaraya sahip gün için gösterge çizilme
   invisible_day=MathRand()%6;
//--- 0 (boş) değer, çizime katılmayacak
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- 0 (boş) değer, çizime katılmayacak
   PlotIndexSetString(0,PLOT_LABEL,"ZigZag1;ZigZag2");
```

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- Çizginin stilini,rengini ve genişliğini değiştirmek için tikleri hesapla
   ticks++;
//--- Yeterli sayıda tik birikmişse
   if(ticks>=N)
     {
      //--- Çizgi özelliklerini değiştir
      ChangeLineAppearance();
      //--- Tik sayacını sıfırla
      ticks=0;
     }

//--- Her bir çubuğun gün bilgisini almak için gereken zaman yapısı
   MqlDateTime dt;

//--- Hesaplamaların başlangıç pozisyonu
   int start=0;
//--- Eğer gösterge bir önceki tikte hesaplanmışsa, hesaplamayı sondan bir önceki çubu
   if(prev_calculated!=0) start=prev_calculated-1;
//--- Hesap döngüsü
   for(int i=start;i<rates_total;i++)
     {
      //--- Yapı içinde çubuk açılış zamanını yaz
      TimeToStruct(time[i],dt);
      //--- Bu çubuğun gün numarası invisible_day değerine eşit
      if(dt.day_of_week==invisible_day)
        {
         //--- Bu çubuk için boş değerleri göstergeye yaz
         ZigZagBuffer1[i]=0;
         ZigZagBuffer2[i]=0;
        }
      //--- Haftanın günü tamamsa, tamponları doldur
```

```
      else
        {
         //--- Çubuk sayısı çift ise
         if(i%2==0)
           {
            //---  İlk tampona High değerini ikinciye Low değerini yaz
            ZigZagBuffer1[i]=high[i];
            ZigZagBuffer2[i]=low[i];
           }
         //--- Sayı teek ise
         else
           {
            //--- Çubuğu ters sırada doldur
            ZigZagBuffer1[i]=low[i];
            ZigZagBuffer2[i]=high[i];
           }
        }
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Zigzag parçalarının görünümünü değiştirir                        |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Zigzag özelliklerine dair bilginin biçimlendirilmesi için bir dizgi
   string comm="";
//--- ZigZag rengini değiştirmek için bir blok
   int number=MathRand(); // Rassal bir sayı al
//--- Bölen, colors[] dizisinin boyutuna eşit
   int size=ArraySize(colors);
//--- Bölümden kalan tam-sayı değeri şeklinde yeni bir renk seçmek için, indis değeri
   int color_index=number%size;
//--- Rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Çizgi rengini ayarla
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Çizgi kalınlığını değiştirmek için bir blok
   number=MathRand();
//--- Tam-sayı bölümden kalan genişliği al
   int width=number%5;   // Genişlik 0'dan 4'e ayarlandı
//--- Rengi PLOT_LINE_WIDTH özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Çizgi kalınlığını yaz
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Çizgi stilini değiştirmek için bir blok
```

```
   number=MathRand();
//--- Bölen, syles dizisinin büyüklüğüne eşit
   size=ArraySize(styles);
//--- Yeni bir stil seçmek için, indis değerini, bölümden kalan tam-sayı değeri şeklir
   int style_index=number%size;
//--- Rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Çizgi stilini yaz
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- Hesaplamada atlanan gün hakkında bilgi ekle
   comm="\r\nNot plotted day - "+EnumToString((ENUM_DAY_OF_WEEK)invisible_day)+comm;
//--- Bir yorum kullanarak, çizelge üzerinde bilgi göster
   Comment(comm);
  }
```

DRAW_FILLING

DRAW_FILLING stili, iki gösterge tamponunun değerleri arasında kalan renkli bir alan çizer. Aslında, bu stilde iki çizgi çizilir, sonra bunların arasında kalan alan, belirtilen iki renkten biri ile doldurulur. Kanal çizen göstergeler için kullanılır. Tamponların hiçbiri baştan sona boş değerler içeremez; böyle bir durumda hiçbir şey çizilmeyecektir.

Iki dolgu rengi ayarlayabilirsiniz:

- ilk renk, birinci tamponun değerlerinin ikincisinden büyük olduğu yerlerde alanı doldurmak için kullanılır;
- ikinci renk ise, ikinci tamponun değerlerinin birinciden büyük olduğu yerlerdeki alanı doldurmak için kullanılır.

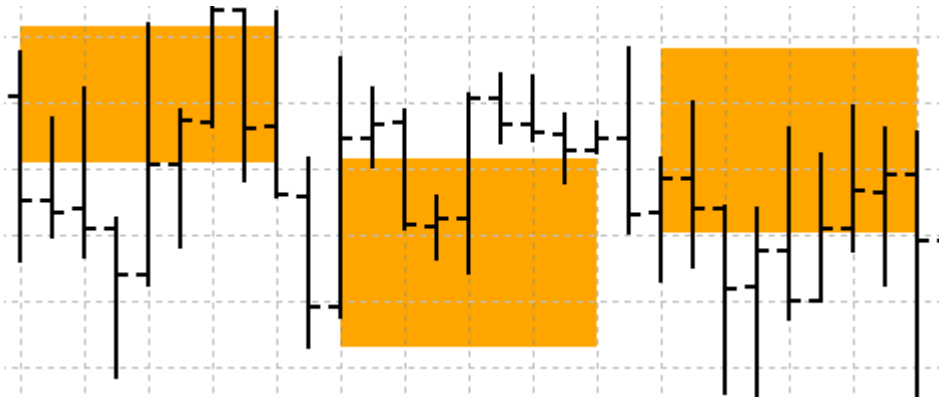
Dolgu rengi, derleyici direktifleri kullanılarak veya dinamik olarak PlotIndexSetInteger() fonksiyonu ile ayarlanabilir. Çizim özelliklerinin dinamik olarak değiştirilmesi, göstergelerin "canlılaştırılmasını" sağlar, bu şekilde göstergelerin görünümleri mevcut duruma göre değişir.

Gösterge, iki gösterge tamponun da 0 veya boş değer içermediği çubuklar için çizilir. Bir değerin "boş" olarak göz önüne alınacağını belirtmek için, bu değeri PLOT_EMPTY_VALUE özelliğiyle ayarlayın:

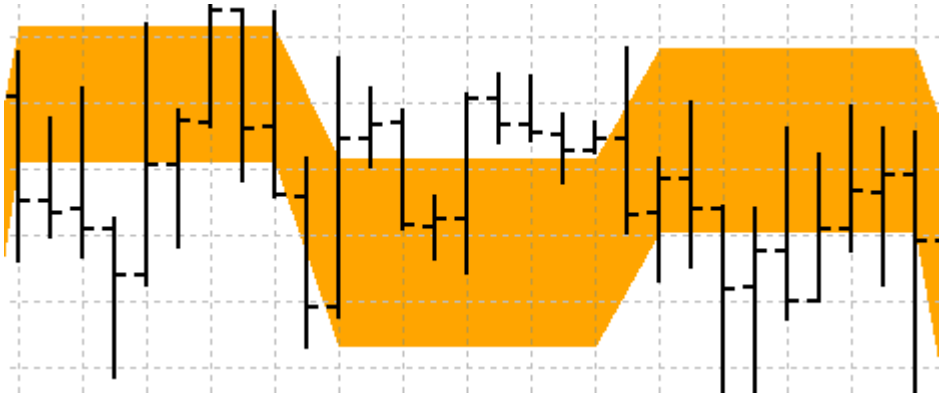
```
   #define INDICATOR_EMPTY_VALUE -1.0
  ...
//--- INDICATOR_EMPTY_VALUE (boş değer) gösterge hesabına katılmayacaktır
   PlotIndexSetDouble (DRAW_FILLING_çizim_indisi,PLOT_EMPTY_VALUE,INDICATOR_EMPTY_VALU
```

Hesaplamaya katılmayan çubuklar üzerinde yapılacak çizim, tamamen gösterge tamponu değerlerine bağlıdır:

- Her iki gösterge tamponunun da 0 olduğu çubuklar gösterge çizimine dahil edilmezler. Yani sıfır değerli alanlar boyanmaz.



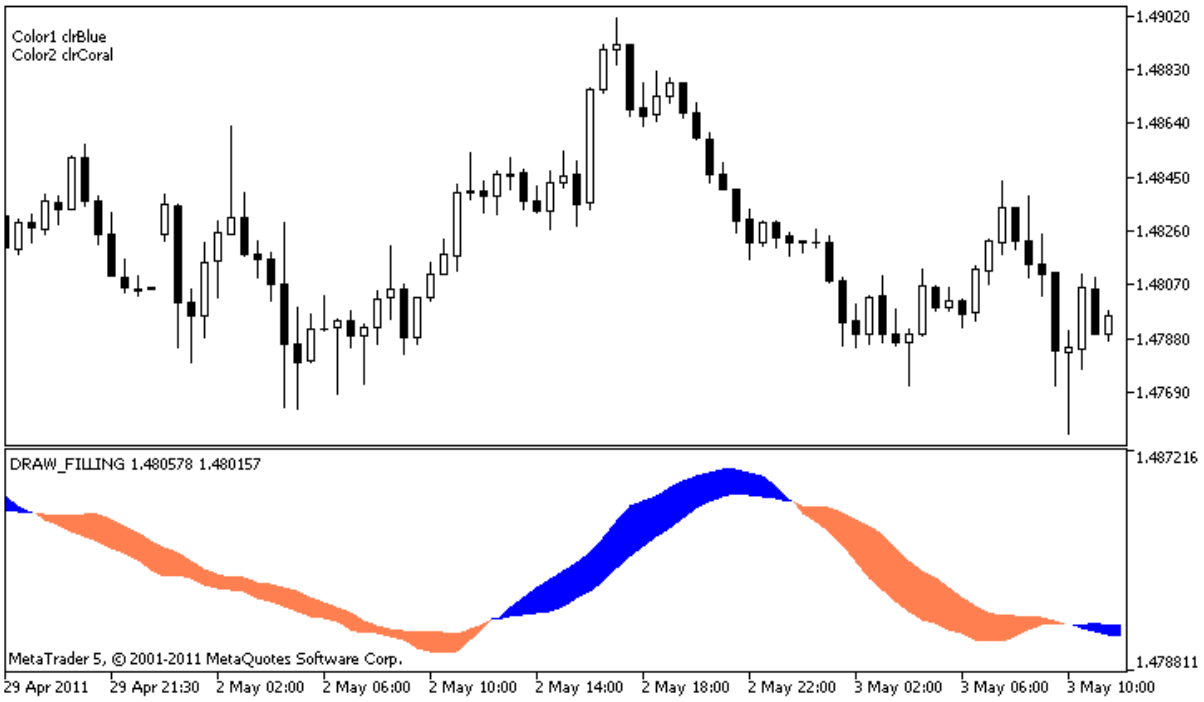
- Diğer taraftan, gösterge tamponlarının "boş değere" sahip olduğu çubuklar da, çizime dahil edilirler. Alanları anlamlı değerlerle birleştirmek için, boş değerli alanlar doldurulacaktır.



Sıfıra eşit olan "boş değerlerin" olması durumunda, gösterge hesabına katılmayan çubukların da doldurulacağı not edilmelidir.

DRAW_FILLING stilinin çizimi için gereken tampon sayısı 2'dir.

Ayrı penceredeki, farklı periyotlara sahip iki Hareketli Ortalamanın (HO) arasına bir kanal çizen gösterge örneği. Hareketli ortalamaların kesişimi sırasındaki renk değişimi, yukarı ve aşağı yönlü trendlerin değişimini görsel olarak betimler. Renk, her **N** tikte bir değişir. N parametresi, el ile yapılandırma olasılığı göz önüne alınarak, göstergenin dışsal parametreleri içinde ayarlanır (göstergenin Özellikler penceresindeki Veriler sekmesi).



DRAW_FILLING stilindeki **plot1** örneğinde, özellikler, #property derleyici direktifi kullanılarak belirlenir, sonrasında ise OnCalculate() fonksiyonunda renkler rassal olarak ayarlanır.

```
//+------------------------------------------------------------------+
//|                                                DRAW_FILLING.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
```

```
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_FILLING stilini betimleyen bir gösterge"
#property description "Ayrı pencerede yer alan iki HO arasına bir kanal çizer"
#property description "dolgu rengi her N tikten sonra"
#property description "rassal olarak değiştirilir"

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot Intersection
#property indicator_label1  "Intersection"
#property indicator_type1   DRAW_FILLING
#property indicator_color1  clrRed,clrBlue
#property indicator_width1  1
//--- giriş parametreleri
input int      Fast=13;          // Hızlı olan ortalamanın periyodu
input int      Slow=21;          // Yavaş olan ortalamanın periyodu
input int      shift=1;          // Ortalamaların ileriye doğru (pozitif) kaydırılması
input int      N=5;              // Değişim için gerekli tik sayısı
//--- Gösterge tamponları
double         IntersectionBuffer1[];
double         IntersectionBuffer2[];
int fast_handle;
int slow_handle;
//--- Renkleri depolayacak bir dizi
color colors[]={clrRed,clrBlue,clrGreen,clrAquamarine,clrBlanchedAlmond,clrBrown,clrCo
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   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);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- Çizginin stilini,rengini ve genişliğini değiştirmek için tikleri hesapla
   ticks++;
//--- Yeterli sayıda tik birikmişse
   if(ticks>=N)
     {
      //--- Çizgi özelliklerini değiştir
      ChangeLineAppearance();
      //--- Tik sayacını sıfırla
      ticks=0;
     }

//--- Göstergenin ilk hesabının yapılması; veya gösterge daha önce hesaplanmışsa, tüm
   if(prev_calculated==0)
     {
      //--- Tüm gösterge değerlerini uygun tamponlara kopyala
      int copied1=CopyBuffer(fast_handle,0,0,rates_total,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,rates_total,IntersectionBuffer2);
     }
   else // Sadece güncellenen veriyi doldur
     {
      //--- OnCalculate() fonksiyonunun şimdiki ve daha önceki çağrıları arasındaki fa
      int to_copy=rates_total-prev_calculated;
      //--- Fark yoksa, sadece tek değer kopyalarız - sıfır çubuğu üzerinde
      if(to_copy==0) to_copy=1;
      //--- to_copy değerlerini gösterge tamponunun en sonuna kopyala
      int copied1=CopyBuffer(fast_handle,0,0,to_copy,IntersectionBuffer1);
      int copied2=CopyBuffer(slow_handle,0,0,to_copy,IntersectionBuffer2);
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Kanal dolgusunun rengini değiştirir                              |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Çizgi özelliklerine dair bilginin biçimlendirilmesi için bir dizgi
   string comm="";
//--- Çizgi rengini değiştirmek için bir blok
```

```
   int number=MathRand(); // Rassal bir sayı al
//--- Bölen, colors[] dizisinin boyutuna eşit
   int size=ArraySize(colors);

//--- Bölümden kalan tam-sayı değeri şeklinde yeni bir renk seçmek için, indis değerir
   int color_index1=number%size;
//--- İlk rengi, PLOT_LINE_COLOR özelliği ile ayarla
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,colors[color_index1]);
//--- İlk rengi yaz
   comm=comm+"\r\nColor1 "+(string)colors[color_index1];

//--- Bölümden kalan tam-sayı değeri şeklinde yeni bir renk seçmek için, indis değerir
   number=MathRand(); // Rassal bir sayı yaz
   int color_index2=number%size;
//--- İkinci rengi, PLOT_LINE_COLOR özelliğiyle ayarla
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,colors[color_index2]);
//--- İkinci rengi yaz
   comm=comm+"\r\nColor2 "+(string)colors[color_index2];
//--- Bir yorum kullanarak, çizelge üzerinde bilgi göster
   Comment(comm);
  }
```

DRAW_BARS

DRAW_BARS stili Open, High, Low ve Close fiyatlarını içeren dört gösterge tamponunun temelinde çubuklar çizer. Bu stil, bir çizelge alt penceresinde yer alanlar ve diğer finansal araçlar da dahil olmak üzere, çubuklar şeklinde göstergeler oluşturulması için tasarlanmıştır.

Çubukların renkleri derleyici direktifleri ile veya dinamik olarak, PlotIndexSetInteger() fonksiyonunu kullanarak ayarlanabilir. Çizim özelliklerinin dinamik olarak değiştirilmesi, göstergelerin "canlılaştırılmasını" sağlar, bu şekilde göstergelerin görünümleri mevcut duruma göre değişir.

Gösterge, dört gösterge tamponundan **hiçbirinin** boş değer içermediği çubuklar için çizilir. Bir değerin "boş" olarak göz önüne alınacağını belirtmek için, bu değeri PLOT_EMPTY_VALUE özelliğine ayarlayın:

```
//--- 0 (boş) değer, çizime katılmayacak
   PlotIndexSetDouble(çizim_indisi_DRAW_BARS,PLOT_EMPTY_VALUE,0);
```

Gösterge tamponlarının değerlerini her zaman açıkça doldurun, atlamak istediğiniz çubukları boş değer olarak ayarlayın.

DRAW_BARS stili için gereken tampon sayısı 4'tür. Çizilecek tüm tamponlar birbiri ardına, verilen sırayla dizilmelidir: Open, High, Low ve Close. Tamponların hiçbiri baştan sona boş değerler içeremez; böyle bir durumda hiçbir şey çizilmeyecektir.

Ayrı bir pencerede yer alan seçilmiş finansal aracın üzerine çubuklar çizen gösterge örneği. Çubukların rengi her **N** tikte rassal olarak değişir. N parametresi, el ile yapılandırma olasılığı göz önüne alınarak, göstergenin dışsal parametreleri içinde ayarlanır (göstergenin Özellikler penceresindeki Veriler sekmesi).

DRAW_BARS stilindeki **plot1** için, renk ve boyut özelliklerinin #property derleyici direktifi kullanılarak belirlendiğini; sonrasında ise rengin, OnCalculate() fonksiyonu içinde önceden hazırlanmış bir listeden rassal olarak ayarlandığını lütfen not ediniz.

```
//+------------------------------------------------------------------+
//|                                                   DRAW_BARS.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_BARS stilini tanıtmak için bir gösterge"
#property description "Seçilen sembolün çubuklarını ayrı bir pencerede çizer"
#property description "Çubukların renk ve genişliği ve ayrıca sembol, her N tikten sor
#property description "rassal olarak değiştirilir"

#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
//--- giriş parametreleri
input int      N=5;              // Tipin değişmesi için gereken tik sayısı
input int      bars=500;         // Gösterilecek çubuk sayısı
input bool     messages=false;   // "Uzmanlar" günlüğünde mesajları göster
//--- Gösterge tamponları
double         BarsBuffer1[];
double         BarsBuffer2[];
double         BarsBuffer3[];
double         BarsBuffer4[];
//--- Sembol ismi
string symbol;
//--- Renkleri depolayacak bir dizi
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Çubuklar çok küçükse - işi planlanandan önce bitir
   if(bars<50)
     {
      Comment("Lüyfen daha fazla sayıda çubuk belirtin! Gösterge işlemi sonlandırıldı"
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,BarsBuffer1,INDICATOR_DATA);
```

```
   SetIndexBuffer(1,BarsBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,BarsBuffer3,INDICATOR_DATA);
   SetIndexBuffer(3,BarsBuffer4,INDICATOR_DATA);
//--- Çubukların çizileceği sembolün ismi
   symbol=_Symbol;
//--- Sembolün görünümünü ayarla
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+symbol+")");
//--- Bir boş değer
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0.0);
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- Çizginin stilini,rengini ve genişliğini değiştirmek için tikleri hesapla
   ticks++;
//--- Yeterli sayıda tik birikmişse
   if(ticks>=N)
     {
      //--- Piyasa Gözlemi penceresinden yeni bir sembol seç
      symbol=GetRandomSymbolName();
      //--- Çizgi özelliklerini değiştir
      ChangeLineAppearance();

      int tries=0;
      //--- Tamponları, sembolün fiyatlarıyla doldurmak için 5 deneme gerçekleştir
      while(!CopyFromSymbolToBuffers(symbol,rates_total) && tries<5)
        {
         //--- CopyFromSymbolToBuffers() fonksiyonunun çağrıları için bir sayaç
         tries++;
        }
      //--- Tik sayacını sıfırla
      ticks=0;
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
```

```
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Gösterge tamponlarını fiyatlarla doldur                          |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total)
  {
//--- rates[] dizisinde, Open, High, Low ve Close fiyatlarını kopyalayacağız
   MqlRates rates[];
//--- Deneme sayacı
   int attempts=0;
//--- Kaç tane kopyalandı
   int copied=0;
//--- Istenen sembole dair bir zaman serisi elde etmek için 25 deneme gerçekleştir
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) denemeler=%d",__FUNCTION__,name,atter
     }
//--- Yeterli sayıda çubuk kopyalanamamışsa
   if(copied!=bars)
     {
      //--- Bir mesaj dizgisinden
      string comm=StringFormat("%s sembolü için, sadece %d çubuk alınabildi (istenen %
                               name,
                               copied,
                               bars
                               );
      //--- Mesajı, ana çizelge penceresindeki bir yorumda göster
      Comment(comm);
      //--- Mesajı göster
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- Sembolün görüntüsünü ayarla
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;"+name+"
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_BARS("+name+")");
     }
//--- Tamponları boş değerlerle başlat
   ArrayInitialize(BarsBuffer1,0.0);
   ArrayInitialize(BarsBuffer2,0.0);
   ArrayInitialize(BarsBuffer3,0.0);
   ArrayInitialize(BarsBuffer4,0.0);
//--- Fiyatları tamponlara kopyala
   for(int i=0;i<copied;i++)
     {
```

```
      //--- Tamponlar için uygun indisleri hesapla
      int buffer_index=total-copied+i;
      //--- Fiyatları tamponlara yaz
      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);
  }
//+------------------------------------------------------------------+
//| Piyasa Gözleminden rassal bir sembole dönüş yapar                |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- Piyasa Gözleminde görünen sembollerin sayısı
   int symbols=SymbolsTotal(true);
//--- Sembolün listedeki pozisyonu - 0'dan symbols'e kadar rassal bir sayı
   int number=MathRand()%symbols;
//--- Belirtilen konumdaki sembolün ismine dönüş yap
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Çubukların görünümünü değiştirir                                |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Çubuk özellikleri ile ilgili bilgiyi biçimlendirmek için bir dizgi
   string comm="";
//--- Çubukların renklerini değiştirmek için bir blok
   int number=MathRand(); // Rassal bir sayı al
//--- Bölen, colors[] dizisinin boyutuna eşit
   int size=ArraySize(colors);
//--- Bölümden kalan tam-sayı değeri şeklinde yeni bir renk seçmek için, indis değeri
   int color_index=number%size;
//--- Rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Çizgi rengini ayarla
   comm=comm+"\r\n"+(string)colors[color_index];

//--- Çubuk genişliğini değiştirmek için bir blok
   number=MathRand();
//--- Tam-sayı bölümden kalan genişliği al
   int width=number%5;   // Genişlik 0'dan 4'e ayarlandı
//--- Rengi PLOT_LINE_WIDTH özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Çizgi kalınlığını yaz
   comm=comm+"\r\nWidth="+IntegerToString(width);
```

```
//--- Sembol ismini yaz
   comm="\r\n"+symbol+comm;

//--- Bir yorum kullanarak, çizelge üzerinde bilgi göster
   Comment(comm);
  }
```

DRAW_CANDLES

DRAW_CANDLES stili Open, High, Low ve Close fiyatlarını içeren dört gösterge tamponunu temel alarak mum şekilli çubuklar çizer. Ayrı bir çizelge alt penceresinde yer alanlar ve diğer finansal araçlar da dahil olmak üzere, mum şeklinde göstergeler oluşturulması için tasarlanmıştır.

Mumların renkleri derleyici direktifleri ile veya dinamik olarak PlotIndexSetInteger() fonksiyonu ile ayarlanabilir. Çizim özelliklerinin dinamik olarak değiştirilmesi göstergelerin "canlılaşmasını" sağlar, bu şekilde göstergelerin görünümleri mevcut duruma göre değişir.

Gösterge, dört gösterge tamponundan **hiçbirinin** boş değer içermediği çubuklar için çizilir. Bir değerin "boş" olarak göz önüne alınacağını belirtmek için, bu değeri PLOT_EMPTY_VALUE bayrağı ile ayarlayın:

```
//--- 0 (boş) değer, çizime katılmayacak
   PlotIndexSetDouble(çizim_indisi_DRAW_CANDLES,PLOT_EMPTY_VALUE,0);
```

Gösterge tamponlarını her zaman açık şekilde doldurun, atlamak istediğiniz çubukları boş değer olarak ayarlayın.

DRAW_CANDLES stili için gereken tampon sayısı 4'tür. Çizilecek tüm tamponlar birbiri ardına ve varsayılan sırayla dizilmelidir: Open, High, Low ve Close. Tamponların hiçbiri baştan sona boş değerler içeremez, zira böyle bir durumda hiçbir şey çizilmeyecektir.

DRAW_CANDLES stilinde mumların görünümünü için üç farklı renk ayarlayabilirsiniz. Sadece bir renk ayarlanması durumunda tüm mumlar belirtilen bu renkle çizilir.

```
//--- tek renk ile çizilen eş-biçiçmli mumlar
#property indicator_label1  "One color candles"
#property indicator_type1   DRAW_CANDLES
//--- sadece bir renk belirtildiği için tüm mumlar aynı renk ile çizilecek
#property indicator_color1  clrGreen
```

Virgülle ayrılmış iki renk belirtilir, birincisi dış-hatlara yani şamdana, ikincisi ise mum gövdesine uygulanır.

```
//--- şamdanlar ve mumlar için farklı renkler
#property indicator_label1  "Two color candles"
#property indicator_type1   DRAW_CANDLES
//--- burada yeşil renk şamdana, beyaz ise mum gövdesine uygulanır
#property indicator_color1  clrGreen,clrWhite
```

Virgülle ayrılmış üç renk belirtilmesi durumunda yükselen ve düşen mumlar farklı renklerle çizilir. Bu durumda ilk renk dış hatlara uygulanırken diğer ikisi sırasıyla boğa ve ayı mumlarına uygulanır.

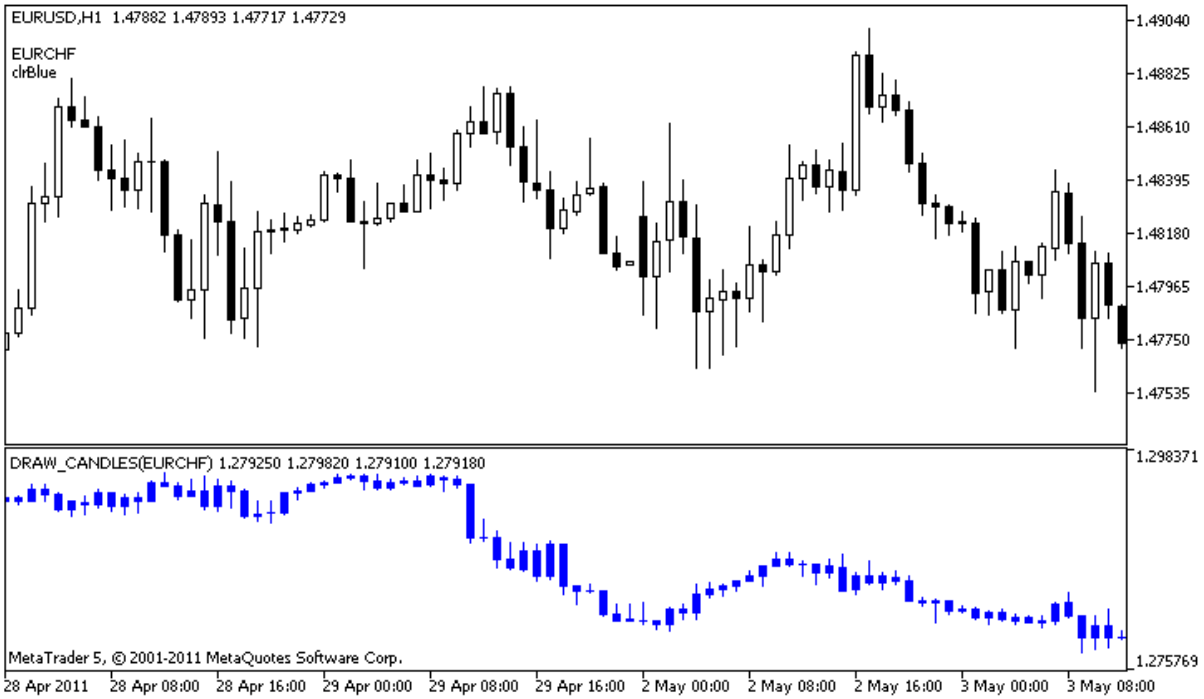
```
//--- şamdanlar ve mumlar için farklı renkler
#property indicator_label1  "One color candles"
#property indicator_type1   DRAW_CANDLES
//--- şamdan rengi yeşil, boğa mumu beyaz, ayı mumu kırmızı
#property indicator_color1  clrGreen,clrWhite,clrRed
```

DRAW_CANDLES stili, bu şekilde özel renklendirme seçenekleri oluşturmanızı sağlar. Ayrıca tüm renkler PlotIndexSetInteger (composition_index_DRAW_CANDLES, PLOT_LINE_COLOR, modifier_index, color) fonksiyonu ile dinamik olarak değiştirilebilir; burada modifier_index şu değerleri alabilir:

- 0 - mumların ve dış hatlarının rengi
- 1- boğa mumunun rengi
- 2 - ayı mumunun rengi

```
//--- mumların ve dış hatlarının rengini ayarla
PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrBlue);
//--- boğa mumunun rengini ayarla
PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrGreen);
//--- ayı mumunun rengini ayarla
PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrRed);
```

Ayrı bir pencerede yer alan seçilmiş finansal aracın üzerine mumlar çizen gösterge örneği. Mumların rengi her **N** tikte rassal olarak değişir. N parametresi el ile yapılandırma olasılığı göz önüne alınarak göstergenin dışsal parametreleri içinde ayarlanır (göstergenin 'Özellikler' penceresindeki 'Veriler' sekmesi).



plot1 için, renk ve boyut özelliklerinin #property derleyici direktifi kullanılarak belirlendiğini; sonrasında ise rengin, OnCalculate() fonksiyonu içinde önceden hazırlanmış bir listeden rassal olarak ayarlandığını lütfen not ediniz.

```
//+------------------------------------------------------------------+
//|                                                  DRAW_CANDLES.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_CANDLES stilini tanıtmak için bir gösterge."
#property description "Seçilen sembolün mumlarını ayrı bir pencerede çizer"
```

```
#property description " "
#property description "Çubukların renk ve genişliği ve sembolü, her N tikten sonra"
#property description "rassal olarak değiştirilir"

#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

//--- giriş parametreleri
input int      N=5;              // Tipin değişmesi için gereken tik sayısı
input int      bars=500;         // Gösterilecek çubuk sayısı
input bool     messages=false;   // "Uzmanlar" günlüğünde mesajları göster
//--- Gösterge tamponları
double         Candle1Buffer1[];
double         Candle1Buffer2[];
double         Candle1Buffer3[];
double         Candle1Buffer4[];
//--- Sembol ismi
string symbol;
//--- Renkleri depolayacak bir dizi
color colors[]={clrRed,clrBlue,clrGreen,clrPurple,clrBrown,clrIndianRed};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Çubuklar çok küçükse işi planlanandan önce bitir
   if(bars<50)
     {
      Comment("Lütfen daha fazla sayıda çubuk belirtin! Gösterge işlemi sonlandırıldı"
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,Candle1Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Candle1Buffer2,INDICATOR_DATA);
   SetIndexBuffer(2,Candle1Buffer3,INDICATOR_DATA);
   SetIndexBuffer(3,Candle1Buffer4,INDICATOR_DATA);
//--- Bir boş değer
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- Çubukların çizileceği sembolün ismi
   symbol=_Symbol;
//--- Sembolün görünümünü ayarla
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
```

```
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_CANDLES("+symbol+")");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- Çizginin stilini,rengini ve genişliğini değiştirmek için tikleri hesapla
   ticks++;
//--- Yeterli sayıda tik birikmişse
   if(ticks>=N)
     {
      //--- Piyasa Gözlemi penceresinden yeni bir sembol seç
      symbol=GetRandomSymbolName();
      //--- Biçimi değiştir
      ChangeLineAppearance();
      //--- Piyasa Gözlemi penceresinden yeni bir sembol seç
      int tries=0;
      //--- plot1 tamponlarını sembolün fiyatlarıyla doldurmak için 5 deneme yap
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            Candle1Buffer1,Candle1Buffer2,Candle1Buffer3,Candle1Buffer4)
            && tries<5)
        {
         //--- CopyFromSymbolToBuffers() fonksiyonunun çağrıları için bir sayaç
         tries++;
        }
      //--- Tik sayacını sıfırla
      ticks=0;
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Belirtilen mumu doldurur                                         |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,
                             int total,
```

```
                              int plot_index,
                              double &buff1[],
                              double &buff2[],
                              double &buff3[],
                              double &buff4[]
                              )
  {
//--- rates[] dizisinde, Open, High, Low ve Close fiyatlarını kopyalayacağız
   MqlRates rates[];
//--- Deneme sayacı
   int attempts=0;
//--- Kaç tane kopyalandı
   int copied=0;
//--- İstenen sembole dair bir zaman serisi elde etmek için 25 deneme gerçekleştir
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) denemeler=%d",__FUNCTION__,name,atter
     }
//--- Yeterli sayıda çubuk kopyalanamamışsa
   if(copied!=bars)
     {
      //--- Bir mesaj dizgisi oluştur
      string comm=StringFormat("%s sembolü için, sadece %d çubuk alınabildi (istenen %
                               name,
                               copied,
                               bars
                               );
      //--- Mesajı ana çizelge penceresindeki bir yorumda göster
      Comment(comm);
      //--- Mesajı göster
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- Sembolün görüntüsünü ayarla
      PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;
     }
//--- Tamponları boş değerlerle başlat
   ArrayInitialize(buff1,0.0);
   ArrayInitialize(buff2,0.0);
   ArrayInitialize(buff3,0.0);
   ArrayInitialize(buff4,0.0);
//--- Her tik ile birlikte fiyatları tamponlara kopyala
   for(int i=0;i<copied;i++)
     {
      //--- Tamponlar için uygun indisleri hesapla
```

```
      int buffer_index=total-copied+i;
      //--- Fiyatları tamponlara yaz
      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);
  }
//+------------------------------------------------------------------+
//| Piyasa Gözleminden rassal bir sembole dönüş yapar                |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- Piyasa Gözleminde görünen sembollerin sayısı
   int symbols=SymbolsTotal(true);
//--- Sembolün listedeki pozisyonu - 0'dan symbols'e kadar rassal bir sayı
   int number=MathRand()%symbols;
//--- Belirtilen konumdaki sembolün ismine dönüş yap
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Çubukların görünümünü değiştirir                                |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Çubuk özellikleri ile ilgili bilgiyi biçimlendirmek için bir dizgi
   string comm="";
//--- Çubukların renklerini değiştirmek için bir blok
   int number=MathRand(); // Rassal bir sayı al
//--- Bölen, colors[] dizisinin boyutuna eşit
   int size=ArraySize(colors);
//--- Bölümden kalan tam-sayı değeri şeklinde yeni bir renk seçmek için indis değerini
   int color_index=number%size;
//--- Rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_COLOR,colors[color_index]);
//--- Rengi yaz
   comm=comm+"\r\n"+(string)colors[color_index];
//--- Sembol ismini yaz
   comm="\r\n"+symbol+comm;
//--- Bir yorum kullanarak çizelge üzerinde bilgi göster
   Comment(comm);
  }
```

DRAW_COLOR_LINE

DRAW_COLOR_LINE stili, DRAW_LINE stilinin renkli versiyonudur. Aynı şekilde bu da gösterge tamponunun değerlerini kullanarak bir çizgi çizer. Ama bu stil, başlığında **COLOR** sözcüğünü taşıyan tüm renkli stiller gibi, özel bir ek tampona sahiptir. Bu tampon, özel olarak ayarlanmış bir renk dizisi içindeki renklerin indislerini depolar. Bu şekilde, her bir çizgi kısmının rengi, söz konusu çubuğun üzerine çizilecek rengin indisinin belirtilmesiyle tanımlanabilir.

Çizgilerin genişlikleri, stilleri ve renkleri, derleyici direktifleri ile veya dinamik olarak PlotIndexSetInteger() fonksiyonunu kullanarak ayarlanabilir. Çizim özelliklerinin dinamik olarak değiştirilmesi, göstergelerin "canlılaştırılmasını" sağlar, bu şekilde göstergelerin görünümleri mevcut duruma göre değişir.

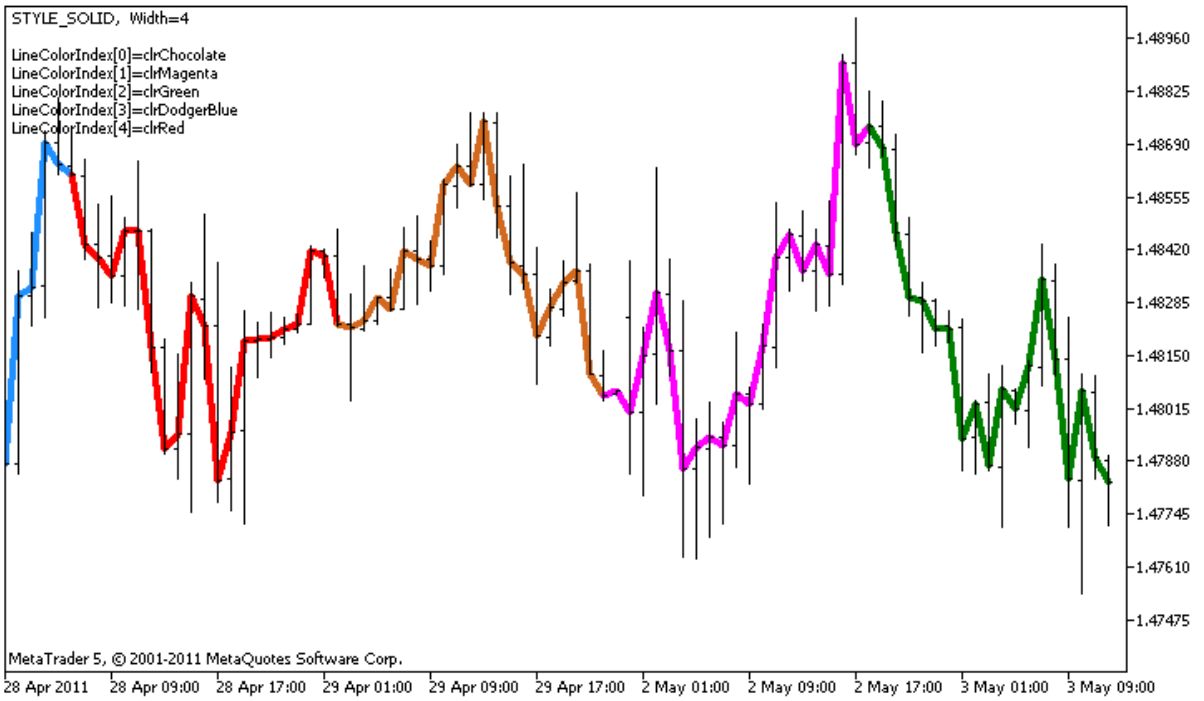
DRAW_COLOR_LINE stilinin çizimi için gereken tampon sayısı 2'dir.

- bir tampon çizginin çizileceği gösterge değerlerini depolamak için;
- bir tampon, çizginin her bir çubuk üzerindeki renk indisini depolamak için.

Renkler, #property indicator_color1 derleyici direktifi ile virgülle ayrılmış şekilde belirtilebilir. Renklerin sayısı 64'ü geçemez.

```
//--- Her bir çubuğu renklendirmek için 5 renk tanımla (özel dizide depolanacaklar)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (en çok 6
```

Çubukların Close (kapanış) fiyatlarını kullanarak çizgi çizen bir gösterge örneği. Çizgi kalınlığı ve stili her N=5 tikte bir rassal olarak değişir.



Çizgi kısımlarının renklerini de, özel ChangeColors() fonksiyonu ile rassal olarak değiştir.

```
//+------------------------------------------------------------------+
//| Çizgi kısımlarının renklerini değiştirir                         |
//+------------------------------------------------------------------+
```

```
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Renklerin sayısı
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Her renk indisi için rassal olarak bir renk tanımla
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Rassal bir değer al
      int number=MathRand();
      //--- col[] dizisi içinde tamsayı bölümünün kalanı şeklinde bir indis al
      int i=number%size;
      //--- Her bir indis için, rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
      PlotIndexSetInteger(0,                    //  Bir grafiksel stilin numarası
                          PLOT_LINE_COLOR,      //  Özellik tanımlayıcı
                          plot_color_ind,       //  Rengin girildiği indis
                          cols[i]);             //  Yeni bir renk
      //--- Renkleri yaz
      comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
```

Örnekte, göstergelerin "renkli" versiyonların özelliği gösterilmektedir - bir çizgi kısmının rengini değiştirmek için (renk indislerini içeren) ColorLineColors[] tamponundaki değerleri değiştirmeniz gerekmez. Tek yapmanız gereken, özel bir dizi içinde yeni renkler ayarlamaktır. Bu, PlotIndexSetInteger() fonksiyonunu ile sadece küçük bir renk dizisini değiştirerek, rengi tüm çizim için hızlıca ve bir kerede değiştirebilmenizi sağlar.

DRAW_COLOR_LINE stilindeki **plot1** için, özelliklerin #property derleyici direktifi kullanılarak belirlendiğini; sonrasında ise üç özelliğin, OnCalculate() fonksiyonu içinde önceden hazırlanmış bir listeden rassal olarak ayarlandığını lütfen not ediniz.

N parametresi ve Length (çubuklardaki renkli kısımların uzunluğu), el ile yapılandırma olasılığı göz önüne alınarak, göstergenin dışsal parametreleri içinde ayarlanır (Gösterge Özellikleri penceresindeki Veriler sekmesi)

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_LINE.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_COLOR_LINE stilinin betimlenmesi için bir gösterge"
```

```
#property description "Close fiyatları üzerinde 20 çubuk üstünde renkli parçalar halir
#property description "çizgi parçalarının renk, genişlik ve stili, her N tikten sonra"
#property description "rassal olarak değiştirilir"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorLine
#property indicator_label1  "ColorLine"
#property indicator_type1   DRAW_COLOR_LINE
//--- Her bir çubuğu renklendirmek için 5 renk tanımla (özel dizide depolanacaklar)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrOrange,clrDeepPink // (en çok 
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- giriş parametreleri
input int      N=5;           // Değişim için kullanılacak tik sayısı
input int      Length=20;     // Çubuklardaki her renk parçasının uzunluğu
int            line_colors=5; // Ayarlanmış renk sayısı 5 - #property indicator_color1
//--- Çizim için bir tampon
double         ColorLineBuffer[];
//--- Herbir çubuk üzerindeki çizgi rengini depolayan bir tampon
double         ColorLineColors[];

//--- Renkleri depolamak için 7 elemanlı bir dizi
color colors[]={clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGolde
//--- Çizgi stillerinin depolanması için bir dizi
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Gösterge tamponunun dizi ile bağlanması
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorLineColors,INDICATOR_COLOR_INDEX);
//--- Pseudo-rassal sayı oluşturucusunun başlatılması
   MathSrand(GetTickCount());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- Çizginin stilini,rengini ve genişliğini değiştirmek için tikleri hesapla
   ticks++;
//--- Tikler önemli bir sayıda biriktirilmişse
   if(ticks>=N)
     {
      //--- Çizgi özelliklerini değiştir
      ChangeLineAppearance();
      //--- Çizgi parçalarının renklerini değiştir
      ChangeColors(colors,5);
      //--- Tik sayacını sıfırla
      ticks=0;
     }

//--- Gösterge değerlerinin hesaplanması için bir blok
   for(int i=0;i<rates_total;i++)
     {
      //--- Gösterge değerini tampona yaz
      ColorLineBuffer[i]=close[i];
      //--- Şimdi, bu çubuk için rassal olarak bir renk ayarla
      int color_index=i%(5*Length);
      color_index=color_index/Length;
      //--- Bu çubuk için çizgi, color_index indisli rengi alacak
      ColorLineColors[i]=color_index;
     }

//--- Fonksiyonun sonraki çağrıları için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Çizgi kısımlarının renklerini değiştirir                         |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Renklerin sayısı
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Her renk indisi için rassal olarak bir renk tanımla
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Rassal bir değer al
      int number=MathRand();
```

```
      //--- col[] dizisi içinde tamsayı bölümünün kalanı şeklinde bir indis al
      int i=number%size;
      //--- Her bir indis için, rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
      PlotIndexSetInteger(0,                    //  Bir grafiksel stilin numarası
                          PLOT_LINE_COLOR,      //  Özellik tanımlayıcı
                          plot_color_ind,       //  Rengin girildiği indis
                          cols[i]);             //  Yeni bir renk
      //--- Renkleri yaz
      comm=comm+StringFormat("LineColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStrinc
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Göstergenin görüntülenen çizgisinin görünümünü değiştirir        |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Çizgi özelliklerine dair bilginin biçimlendirilmesi için bir dizgi
   string comm="";
//--- Çizgi kalınlığını değiştirmek için bir blok
   int number=MathRand();
//--- Tam-sayı bölümden kalan genişliği al
   int width=number%5; // Genişlik 0'dan 4'e ayarlanır
//--- Rengi PLOT_LINE_WIDTH özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Çizgi kalınlığını yaz
   comm=comm+" Width="+IntegerToString(width);

//--- Çizgi stilini değiştirmek için bir blok
   number=MathRand();
//--- Bölen, syles dizisinin büyüklüğüne eşit
   int size=ArraySize(styles);
//--- Yeni bir stil seçmek için, indis değerini, bölümden kalan tam-sayı değeri şeklir
   int style_index=number%size;
//--- Rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Çizgi stilini yaz
   comm=EnumToString(styles[style_index])+", "+comm;
//--- Bir yorum kullanarak, çizelge üzerinde bilgi göster
   Comment(comm);
  }
```

DRAW_COLOR_SECTION

DRAW_COLOR_SECTION stili, DRAW_SECTION stilinin renkli versiyonudur ama onun aksine, farklı renkteki kısımların çizimine olanak sağlar. DRAW_COLOR_SECTION stili, başlığında **COLOR** sözcüğünü taşıyan tüm renkli stiller gibi, özel bir ek tampona sahiptir. Bu tampon, özel olarak ayarlanmış bir renk dizisi içindeki renklerin indislerini depolar. Bu şekilde, her bir çizgi parçasının rengi, parçanın sonuna denk gelen çubuğun üzerine çizilecek renk indisinin belirtilmesiyle tanımlanabilir.

Parçaların kalınlık, renk ve stilleri, DRAW_SECTION stiline benzer şekilde, derleyici direktifleri ile veya dinamik olarak PlotIndexSetInteger() fonksiyonunun kullanımıyla belirtilebilir. Çizim özelliklerinin dinamik olarak değiştirilmesi, göstergelerin "canlılaştırılmasını" sağlar, bu şekilde göstergelerin görünümleri mevcut duruma göre değişir.

Parçalar (kısımlar), gösterge tamponunun boş olmayan bir değerinden, yine boş olmayan bir değerine kadar çizilir ve boş değerler gözardı edilir. Bir değerin "boş" olarak göz önüne alınacağını belirtmek için, bu değeri PLOT_EMPTY_VALUE özelliği ile ayarlayın: Örneğin, gösterge, sıfır olmayan değerler üzerindeki parçalardan oluşan bir dizi şeklinde çizilecekse, sıfır değerini boş değer olarak ayarlamalısınız:

```
//--- 0 (boş) değer, çizime katılmayacak
   PlotIndexSetDouble(çizim_indisi_DRAW_COLOR_SECTION,PLOT_EMPTY_VALUE,0);
```

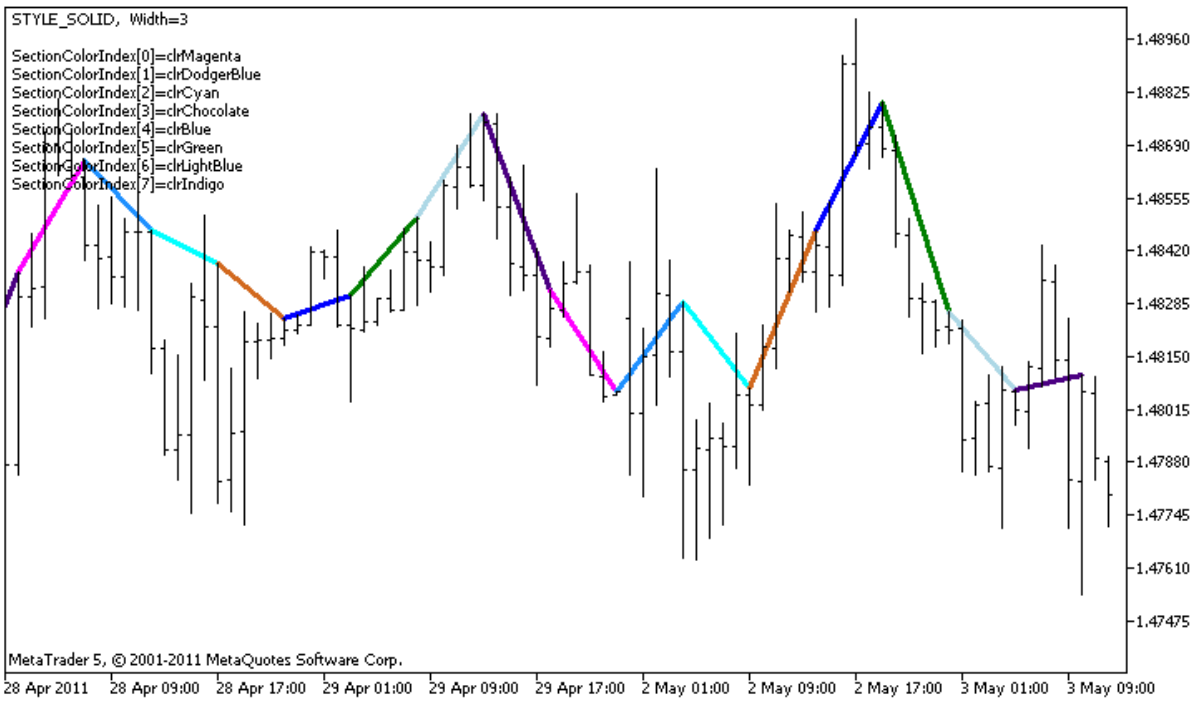
Gösterge tamponlarının değerlerini her zaman açıkça doldurun, atlamak istediğiniz çubukları boş değer olarak ayarlayın.

DRAW_COLOR_SECTION tamponunun çizimi için gereken tamponların sayısı 2'dir.

- bir tampon çizginin çizileceği gösterge değerlerini depolamak için;
- bir tampon, parçayı çizmekte kullanılan renk indisini depolamak için (sadece, boş olmayan değerlerler için ayarlanması mantıklıdır).

Renkler, #property indicator_color1 derleyici direktifi ile virgülle ayrılmış şekilde belirtilebilir. Renklerin sayısı 64'ü geçemez.

High (yüksek) fiyat değerlerini kullanarak, 5 çubuk uzunluğunda renkli parçalar çizen bir gösterge örneği. Parçaların renkleri, kalınlıkları ve stilleri **N** tikte bir rassal olarak değişir.



DRAW_COLOR_SECTION stilindeki **plot1** için, özelliklerin başlangıçta #property derleyici direktifi kullanılarak belirlendiğini; Sonrasında ise renklerin, OnCalculate() fonksiyonu içinde önceden hazırlanmış bir listeden rassal olarak ayarlandığını lütfen not ediniz.

N parametresi, el ile yapılandırma olasılığı göz önüne alınarak, göstergenin dışsal parametreleri içinde ayarlanır (Gösterge Özellikleri penceresindeki Veriler sekmesi).

```
//+------------------------------------------------------------------+
//|                                            DRAW_COLOR_SECTION.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_COLOR_SECTION stilini betimleyen bir gösterge"
#property description "Belirtilen çubuk sayısı uzunluğunda renkli parçalar çizer"
#property description "Parçaların renk, genişlik ve stilleri, her N tikten sonra"
#property description "rassal olarak değiştirilir"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorSection
#property indicator_label1  "ColorSection"
#property indicator_type1   DRAW_COLOR_SECTION
//--- Kısımları renklendirmek için 8 renk tanımla (özel dizide depolanırlar)
#property indicator_color1  clrRed,clrGold,clrMediumBlue,clrLime,clrMagenta,clrBrown,c
#property indicator_style1  STYLE_SOLID
```

```
#property indicator_width1  1
//--- giriş parametreleri
input int      N=5;                      // Değişimin gerçekleştirileceği tik sayısı
input int      bars_in_section=5;        // Çubuk sayısı olarak, parçaların uzunluğu
//--- Parçaların sonlarını hesaplamak için ek bir değişken
int            divider;
int            color_sections;
//--- Çizim için bir tampon
double         ColorSectionBuffer[];
//--- Herbir çubuk üzerindeki çizgi rengini depolayan bir tampon
double         ColorSectionColors[];
//--- Renkleri depolayacak 14 elemanlı bir dizi
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- Çizgi stillerinin depolanması için bir dizi
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,ColorSectionBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorSectionColors,INDICATOR_COLOR_INDEX);
//--- 0 (boş) değer, çizime katılmayacak
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- Parçaların renklendirilmesinde kullanılacak renk sayısı
   int color_sections=8;   //  #property indicator_color1 özelliğinde yoruma bakın
//--- Gösterge parametresini kontrol et
   if(bars_in_section<=0)
     {
      PrintFormat("Geçersiz parça uzunluğu=%d",bars_in_section);
      return(INIT_PARAMETERS_INCORRECT);
     }
   else divider=color_sections*bars_in_section;
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- Çizginin stilini,rengini ve genişliğini değiştirmek için tikleri hesapla
   ticks++;
//--- Tikler önemli bir sayıda biriktirilmişse
   if(ticks>=N)
     {
      //--- Çizgi özelliklerini değiştir
      ChangeLineAppearance();
      //--- Parçaların çiziminde kullanılan renkleri değiştir
      ChangeColors(colors,color_sections);
      //--- Tik sayacını sıfırla
      ticks=0;
     }

//--- Gösterge değerlerinin hesaplanmaya başlanacağı çubuğun numarası
   int start=0;
//--- Gösterge daha önce hesaplanmışsa, başlangıcı bir önceki çubuğa al
   if(prev_calculated>0) start=prev_calculated-1;
//--- Burada göstergenin tüm değerleri hesaplanır
   for(int i=start;i<rates_total;i++)
     {
      //--- Eğer çubuk numarası section_length ile bölünebiliyorsa, parçanın sonuna ge
      if(i%bars_in_section==0)
        {
         //--- Parçanın sonunu bu çubuğun High (yüksek) fiyatına ayarla
         ColorSectionBuffer[i]=high[i];
         //--- Çubuk sayısının scetion_length*number_of_colors değerine bölümünden kal
         int rest=i%divider;
         //renk numarasını =  0'dan number_of_colors-1 değerine kadar al
         int color_indext=rest/bars_in_section;
         ColorSectionColors[i]=color_indext;
        }
      //---Bölümden kalan 'bars' değerine eşitse,
      else
        {
         //--- Hiçbir değişim olmadıysa çubuğu gözardı et - 0 olarak ayarla
         else ColorSectionBuffer[i]=0;
        }
     }
//--- Fonksiyonun sonraki çağrıları için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

```
//| Çizgi kısımlarının renklerini değiştirir                         |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Renklerin sayısı
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Her renk indisi için rassal olarak bir renk tanımla
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Rassal bir değer al
      int number=MathRand();
      //--- col[] dizisi içinde tamsayı bölümünün kalanı şeklinde bir indis al
      int i=number%size;
      //--- Her bir indis için, rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
      PlotIndexSetInteger(0,                    //  Bir grafiksel stilin numarası
                          PLOT_LINE_COLOR,      //  Özellik tanımlayıcı
                          plot_color_ind,       //  Rengin girildiği indis
                          cols[i]);             //  Yeni bir renk
      //--- Renkleri yaz
      comm=comm+StringFormat("SectionColorIndex[%d]=%s \r\n",plot_color_ind,ColorToSt
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Göstergenin görüntülenen çizgisinin görünümünü değiştirir        |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Çizgi özelliklerine dair bilginin biçimlendirilmesi için bir dizgi
   string comm="";
//--- Çizgi kalınlığını değiştirmek için bir blok
   int number=MathRand();
//--- Tam-sayı bölümden kalan genişliği al
   int width=number%5; // Genişlik 0'dan 4'e ayarlanır
//--- Rengi PLOT_LINE_WIDTH özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Çizgi kalınlığını yaz
   comm=comm+" Width="+IntegerToString(width);

//--- Çizgi stilini değiştirmek için bir blok
   number=MathRand();
//--- Bölen, syles dizisinin büyüklüğüne eşit
   int size=ArraySize(styles);
//--- Yeni bir stil seçmek için, indis değerini, bölümden kalan tam-sayı değeri şeklir
   int style_index=number%size;
```

```
//--- Rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Çizgi stilini yaz
   comm=EnumToString(styles[style_index])+", "+comm;
//--- Bir yorum kullanarak, çizelge üzerinde bilgi göster
   Comment(comm);
  }
```

DRAW_COLOR_HISTOGRAM

DRAW_COLOR_HISTOGRAM stili, sıfırdan belirtilen bir değere kadar bir renkli sütun dizisi şeklinde histogram çizer. Değerler gösterge tamponundan alınır. Her bir sütun, daha önceden tanımlan mış bir küme içerisinden, kendine has bir renge sahip olabilir.

Histogramın kalınlık, renk ve stilleri, DRAW_HISTOGRAM stiline benzer şekilde, derleyici direktifleri ile veya dinamik olarak PlotIndexSetInteger() fonksiyonunun kullanımıyla belirtilebilir. Çizim özelliklerinin dinamik olarak değişmesi, mevcut duruma bağlı olarak, histogram görünümünün de değişmesini sağlar.

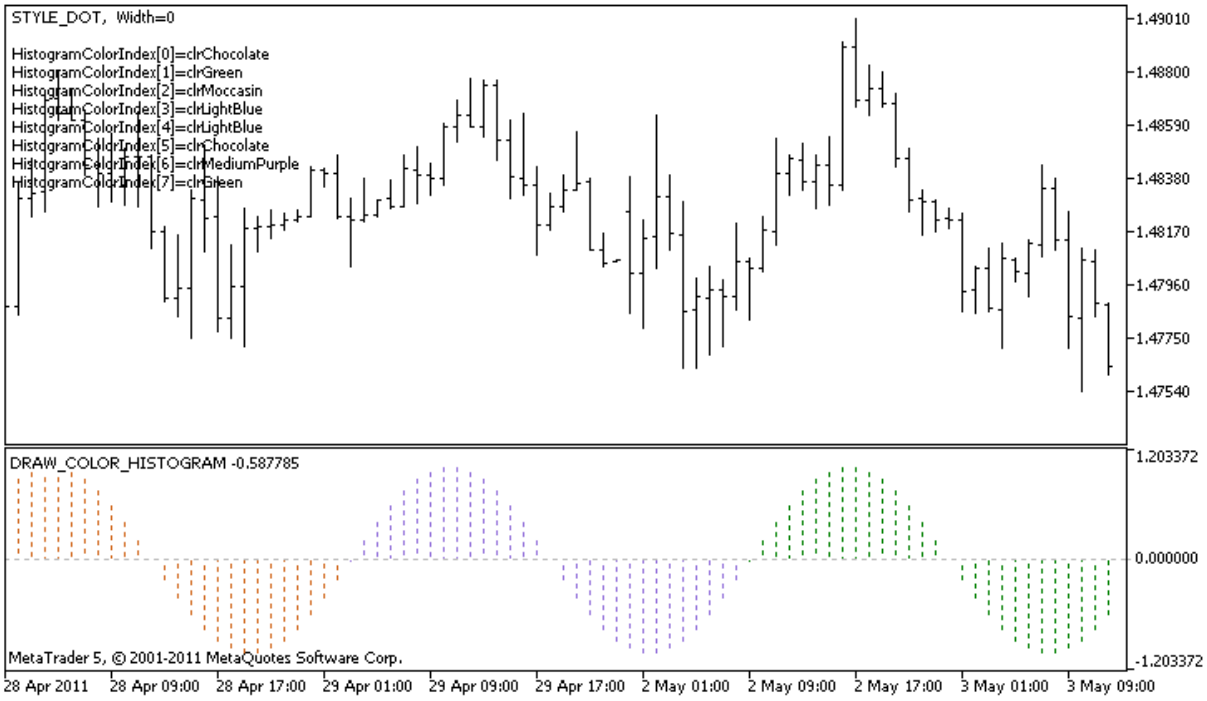
Her çubuk için sıfır seviyesinden bir sütun çizildiğinden, DRAW_COLOR_HISTOGRAM stilinin ayrı bir pencerede kullanılması daha iyi olacaktır. Bu çizim tipi genellikle osilatör tipi göstergeler çizmek için kullanılır, örneğin, Awesome Oscillator veya Market Facilitation Index. Görüntülenmeyecek boş değerler için sıfır değeri belirtilmelidir.

DRAW_COLOR_HISTOGRAM stilinin çizimi için gereken tampon sayısı 2'dir.

- bir tampon, her çubuktaki dikey kısmın sıfır olmayan değerlerinin depolanması için kullanılır; kısımın ikinci ucu, her zaman göstergenin sıfır çizgisi üzerinde olur;
- bir tampon, parçayı çizmekte kullanılan renk indisini depolamak için (sadece, boş olmayan değerlerler için ayarlanması mantıklıdır).

Renkler, virgülle ayrılmış şekilde #property indicator_color1 derleyici direktifi ile belirlenebilir. Renklerin sayısı 64'ü geçemez.

MathSin() fonksiyonu temelinde, belirtilen bir renk ile bir sinüsoid çizen gösterge örneği. **Tüm** histogram sütunlarının renkleri, genişlikleri ve stilleri, her **N** tikte bir rassal olarak değişir. "bars" parametresi, sinüsoidin periyodunu belirtir, yani belirtilen sayıdaki çubuğun ardından sinüsoid döngüyü tekrar edecektir.



DRAW_COLOR_HISTOGRAM stiline sahip **plot1** için, #property derleyici direktifi ile başlangıçta 5 renk ayarlandığını, daha sonra OnCalculate() fonksiyonu içerisinde rengin colors[] dizisinde depolanan 14 renk arasından rassal olarak seçildiğini lütfen not ediniz. N parametresi, el ile yapılandırma olasılığı göz önüne alınarak, göstergenin dışsal parametreleri içinde ayarlanır (göstergenin özellikler penceresindeki Veriler sekmesi).

```
//+------------------------------------------------------------------+
//|                                          DRAW_COLOR_HISTOGRAM.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_COLOR_HISTOGRAM stilini betimleyen bir gösterge"
#property description "Ayrı bir pencerede, histogram şeklinde bir sinüsoid çizer"
#property description "Çubukların renk ve genişliği, her N tikten sonra"
#property description "rassal olarak değiştirilir"
#property description "bars parametresi, sinüsoidin tekrar edeceği çubuk sayısını ayar

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- giriş parametreleri
input int      bars=30;          // Çubuk sayısıyla, sinüsoid periyodu
input int      N=5;              // Histogramı değiştirmek için gereken tik sayısı
//--- plot Color_Histogram
#property indicator_label1  "Color_Histogram"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
//--- Kısımları renklendirmek için 8 renk tanımla (özel dizide depolanırlar)
#property indicator_color1  clrRed,clrGreen,clrBlue,clrYellow,clrMagenta,clrCyan,clrMe
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Değerler için bir tampon
double         Color_HistogramBuffer[];
//--- İndisler için bir tampon
double         Color_HistogramColors[];
//--- Bars parametresi ile çarpıldığında, 2Pi'lik açıyı radyan olarak alabilmemiz için
double         multiplier;
int            color_sections;
//--- Renkleri depolayacak 14 elemanlı bir dizi
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- Çizgi stillerinin depolanması için bir dizi
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
```

```
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,Color_HistogramBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,Color_HistogramColors,INDICATOR_COLOR_INDEX);
//---- Sinüsoidin renklendirilmesi için gereken renklerin sayısı
   color_sections=8;   // #property indicator_color1 özelliğindeki yoruma bak
//--- Çarpan değerini hesapla
   if(bars>1)multiplier=2.*M_PI/bars;
   else
     {
      PrintFormat("bars=%d değerini 1'den büyük olarak ayarla",bars);
      //--- Göstergenin erken sonlandırılması
      return(INIT_PARAMETERS_INCORRECT);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- Çizginin stilini,rengini ve genişliğini değiştirmek için tikleri hesapla
   ticks++;
//--- Tikler önemli bir sayıda biriktirilmişse
   if(ticks>=N)
     {
      //--- Çizgi özelliklerini değiştir
      ChangeLineAppearance();
      //--- Histogram için kullanılan renkleri değiştir
      ChangeColors(colors,color_sections);
      //--- Tik sayacını sıfırla
      ticks=0;
     }
```

```
//--- Gösterge değerlerini hesapla
   int start=0;
//--- Daha önceki OnCalculate çağrısında hesaplanmışsa
   if(prev_calculated>0) start=prev_calculated-1; // hesaplama başlangıcını son seferi
//--- Gösterge tamponunu değerlerle doldur
   for(int i=start;i<rates_total;i++)
     {
      //--- Bir değer
      Color_HistogramBuffer[i]=sin(i*multiplier);
      //--- Renk
      int color_index=i%(bars*color_sections);
      color_index/=bars;
      Color_HistogramColors[i]=color_index;
     }
//--- Fonksiyonun sonraki çağrıları için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Çizgi kısımlarının renklerini değiştirir                         |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Renklerin sayısı
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Her renk indisi için rassal olarak bir renk tanımla
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Rassal bir değer al
      int number=MathRand();
      //--- col[] dizisi içinde tamsayı bölümünün kalanı şeklinde bir indis al
      int i=number%size;
      //--- Her bir indis için, rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
      PlotIndexSetInteger(0,                    //  Bir grafiksel stilin numarası
                          PLOT_LINE_COLOR,      //  Özellik tanımlayıcı
                          plot_color_ind,       //  Rengin girildiği indis
                          cols[i]);             //  Yeni bir renk
      //--- Renkleri yaz
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToS
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Göstergenin görüntülenen çizgisinin görünümünü değiştirir        |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
```

```
  {
//--- Çizgi özelliklerine dair bilginin biçimlendirilmesi için bir dizgi
   string comm="";
//--- Çizgi kalınlığını değiştirmek için bir blok
   int number=MathRand();
//--- Tam-sayı bölümden kalan genişliği al
   int width=number%5; // Genişlik 0'dan 4'e ayarlanır
//--- Rengi PLOT_LINE_WIDTH özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Çizgi kalınlığını yaz
   comm=comm+" Width="+IntegerToString(width);

//--- Çizgi stilini değiştirmek için bir blok
   number=MathRand();
//--- Bölen, syles dizisinin büyüklüğüne eşit
   int size=ArraySize(styles);
//--- Yeni bir stil seçmek için, indis değerini, bölümden kalan tam-sayı değeri şeklin
   int style_index=number%size;
//--- Rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Çizgi stilini yaz
   comm=EnumToString(styles[style_index])+", "+comm;
//--- Bir yorum kullanarak, çizelge üzerinde bilgi göster
   Comment(comm);
  }
```

DRAW_COLOR_HISTOGRAM2

DRAW_COLOR_HISTOGRAM2 stili, belirtilen renkte bir histogram çizer - iki gösterge değerinin değerlerini kullanarak dikey kısımları çizer. DRAW_HISTOGRAM2 stilinin aksine, bu stilde her bir sütun, daha önceden tanımlan mış bir küme içerisinden, kendine has bir renge sahip olabilir. Tüm kısımların değerleri gösterge tamponundan alınır.

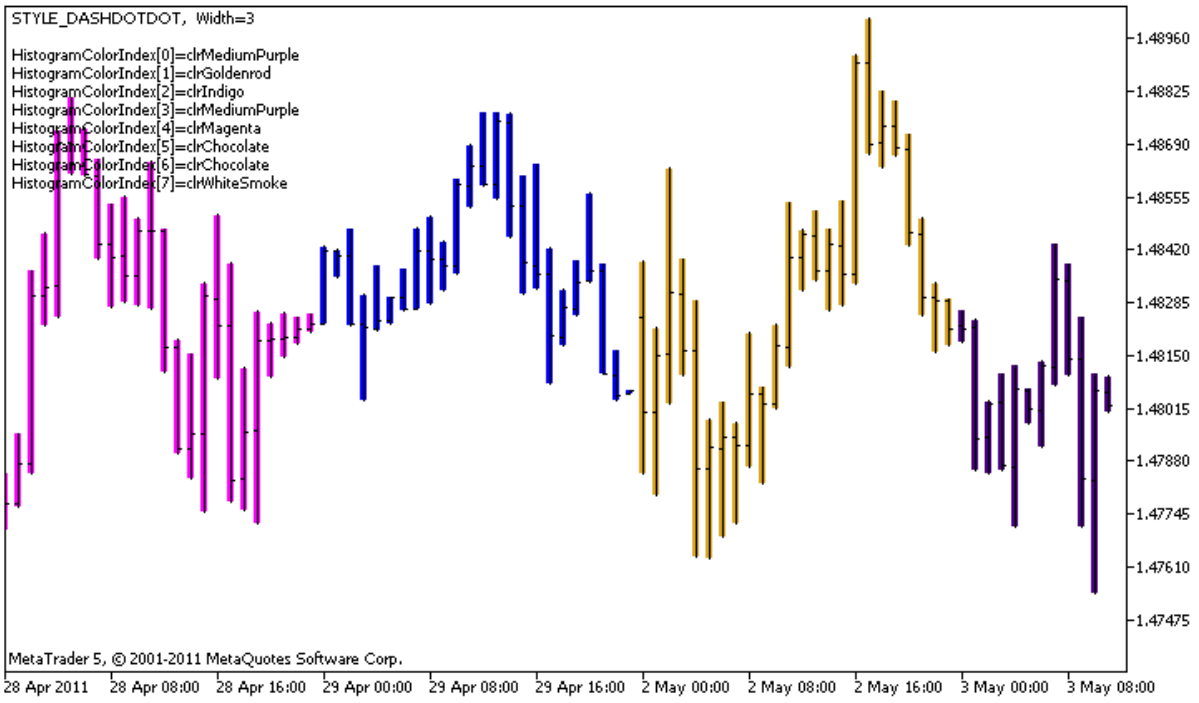
Sembollerin kalınlık ve genişliklerDRAW_HISTOGRAM2 stiline benzer şekilde, derleyici direktifleri ile veya dinamik olarak PlotIndexSetInteger() fonksiyonunun kullanımıyla belirtilebilir. Çizim özelliklerinin dinamik olarak değişmesi, mevcut duruma bağlı olarak, histogram görünümünün de değişmesini sağlar.

DRAW_COLOR_HISTOGRAM2 stili, çizelge ana penceresinde veya ayrı alt pencerelerde kullanılabilir. Boş değerler için hiçbir şey çizilmez; gösterge tamponundaki tüm değerler açıkça ayarlanmalıdır. Tamponlar boş değerler ile başlatılmazlar.

DRAW_COLOR_HISTOGRAM2 stilinin çizimi için gereken tampon sayısı 3'tür:

- iki tampon, çubuk üzerindeki dikey kısımların üst ve alt sınırlarını depolamak için;
- bir tampon, kısımı çizmekte kullanılacak rengin indisini depolamak için (sadece, boş olmayan değerlerler için ayarlanması mantıklıdır).

High (yüksek) ve Low (düşük) fiyatlar arasında, belirtilen renkte bir histogram çizen gösterge örneği. Histogram, haftanın her günü için ayrı bir değer alır. Histogramın gün rengi, kalınlığı ve stili, her **N** tik ile rassal olarak değişir.



DRAW_COLOR_HISTOGRAM2 stiline sahip **plot1** için, #property derleyici direktifi ile başlangıçta 5 renk ayarlandığını, daha sonra rengin, OnCalculate() fonksiyonu içerisindeki colors[] dizisinde depolanan 14 renk arasından rassal olarak seçildiğini lütfen not ediniz.

N parametresi, el ile yapılandırma olasılığı göz önüne alınarak, göstergenin dışsal parametreleri içinde ayarlanır (Gösterge Özellikleri penceresindeki Veriler sekmesi).

```
//+------------------------------------------------------------------+
//|                                          DRAW_COLOR_HISTOGRAM2.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_COLOR_HISTOGRAM2 stilini betimleyen bir gösterge"
#property description "Her bir çubuk üzerinde Open ve Close arasındaki bir kısımı çize
#property description "renk, genişlik ve stil, her N tikten sonra"
#property description "rassal olarak değiştirilir"

#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
//--- Haftanın günlerine göre histogramı renklendirmek için 5 renk tanımla (özel dizi
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- giriş parametreleri
input int      N=5;              // Histogramı değiştirmek için gereken tik sayısı
int            color_sections;
//--- Değer tamponları
double         ColorHistogram_2Buffer1[];
double         ColorHistogram_2Buffer2[];
//--- İndisler için bir tampon
double         ColorHistogram_2Colors[];
//--- Renkleri depolayacak 14 elemanlı bir dizi
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- Çizgi stillerinin depolanması için bir dizi
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,ColorHistogram_2Buffer1,INDICATOR_DATA);
   SetIndexBuffer(1,ColorHistogram_2Buffer2,INDICATOR_DATA);
```

```
   SetIndexBuffer(2,ColorHistogram_2Colors,INDICATOR_COLOR_INDEX);
//--- Bir boş değer ayarla
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- Sinüsoidin renklendirilmesi için gereken renklerin sayısı
   color_sections=8;   //  #property indicator_color1 özelliğindeki yoruma bak
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- Çizginin stilini,rengini ve genişliğini değiştirmek için tikleri hesapla
   ticks++;
//--- Tikler önemli bir sayıda biriktirilmişse
   if(ticks>=N)
     {
      //--- Çizgi özelliklerini değiştir
      ChangeLineAppearance();
      //--- Histogramın çiziminde kullanılacak renkleri değiştir
      ChangeColors(colors,color_sections);
      //--- Tik sayacını sıfırla
      ticks=0;
     }

//--- Gösterge değerlerini hesapla
   int start=0;
//--- Haftanın günlerini her bir çubuğun açılışını kullanarak almak için
   MqlDateTime dt;
//--- Daha önceki OnCalculate çağrısında hesaplanmışsa
   if(prev_calculated>0) start=prev_calculated-1; // hesaplama başlangıcını son seferi
//--- Gösterge tamponunu değerlerle doldur
   for(int i=start;i<rates_total;i++)
     {
      TimeToStruct(time[i],dt);
      //--- değer
      ColorHistogram_2Buffer1[i]=high[i];
      ColorHistogram_2Buffer2[i]=low[i];
```

```
      //--- Haftanın gününe göre renk indisini ayarla
      int day=dt.day_of_week;
      ColorHistogram_2Colors[i]=day;
     }
//--- Fonksiyonun sonraki çağrıları için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Çizgi kısımlarının renklerini değiştirir                         |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Renklerin sayısı
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Her renk indisi için rassal olarak bir renk tanımla
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Rassal bir değer al
      int number=MathRand();
      //--- col[] dizisi içinde, tam-sayı bölümünden kalan şeklinde bir indisi al
      int i=number%size;
      //--- Her bir indis için, rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
      PlotIndexSetInteger(0,                    //  Bir grafiksel stilin numarası
                          PLOT_LINE_COLOR,      //  Özellik tanımlayıcı
                          plot_color_ind,       //  Rengin girildiği indis
                          cols[i]);             //  Yeni bir renk
      //--- Renkleri yaz
      comm=comm+StringFormat("HistogramColorIndex[%d]=%s \r\n",plot_color_ind,ColorToS
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Göstergenin görüntülenen çizgisinin görünümünü değiştirir        |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Çizgi özelliklerine dair bilginin biçimlendirilmesi için bir dizgi
   string comm="";
//--- Çizgi kalınlığını değiştirmek için bir blok
   int number=MathRand();
//--- Tam-sayı bölümden kalan genişliği al
   int width=number%5; // Genişlik 0'dan 4'e ayarlanır
//--- Rengi PLOT_LINE_WIDTH özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Çizgi kalınlığını yaz
```

```
   comm=comm+" Width="+IntegerToString(width);

//--- Çizgi stilini değiştirmek için bir blok
   number=MathRand();
//--- Bölen, syles dizisinin büyüklüğüne eşit
   int size=ArraySize(styles);
//--- Yeni bir stil seçmek için, indis değerini, bölümden kalan tam-sayı değeri şeklir
   int style_index=number%size;
//--- Rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Çizgi stilini yaz
   comm=EnumToString(styles[style_index])+", "+comm;
//--- Bir yorum kullanarak, çizelge üzerinde bilgi göster
   Comment(comm);
  }
```

DRAW_COLOR_ARROW

DRAW_COLOR_ARROW stili, gösterge tamponunun değerlerine göre renkli oklar çizer, (Wingdings tipi semboller). DRAW_ARROW stilinin aksine bu stilde her bir sembolün rengi, indicator_color1 özelliği kullanılarak, önceden tanımlanmış bir dizi renk içinden seçilebilir.

Sembollerin kalınlık ve genişlikleriDRAW_ARROW stiline benzer şekilde, derleyici direktifleri ile veya dinamik olarak PlotIndexSetInteger() fonksiyonunun kullanımıyla belirtilebilir. Çizim özelliklerinin dinamik olarak değişmesi, mevcut duruma bağlı olarak, gösterge görünümünün değişmesini sağlar.

Sembol kodu, PLOT_ARROW özelliği kullanılarak ayarlanır.

```
//--- PLOT_ARROW'da çizmek için Wingdings yazı tipinden bir sembolün kodunu tanımla
   PlotIndexSetInteger(0,PLOT_ARROW,code);
```

PLOT_ARROW için varsayılan değer 159'dur (daire).

Her ok, gerçekte bir yüksekliğe ve tutturma noktasına sahip olan bir semboldür ve çizelge üzerinde bazı önemli bilgileri örtebilir (örneğin bir çubuğun kapanış fiyatı). Bu nedenle, çizelge ölçeğine bağlı olmayan, piksel bazında bir dikey kaydırmayı ayrıyeten belirtebiliriz. Bu durumda oklar, belirtilen piksel sayısı kadar aşağı kaydırılacaktır, ayrıca gösterge değeri de aynı kalacaktır:

```
//--- Oklar için piksel bazında dikey kaydırma ayarla
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
```

PLOT_ARROW_SHIFT'in negatif değerleri, okun yukarı kaydırılacağı anlamına gelir, pozitif bir değer ise aşağı kaydırma demektir.

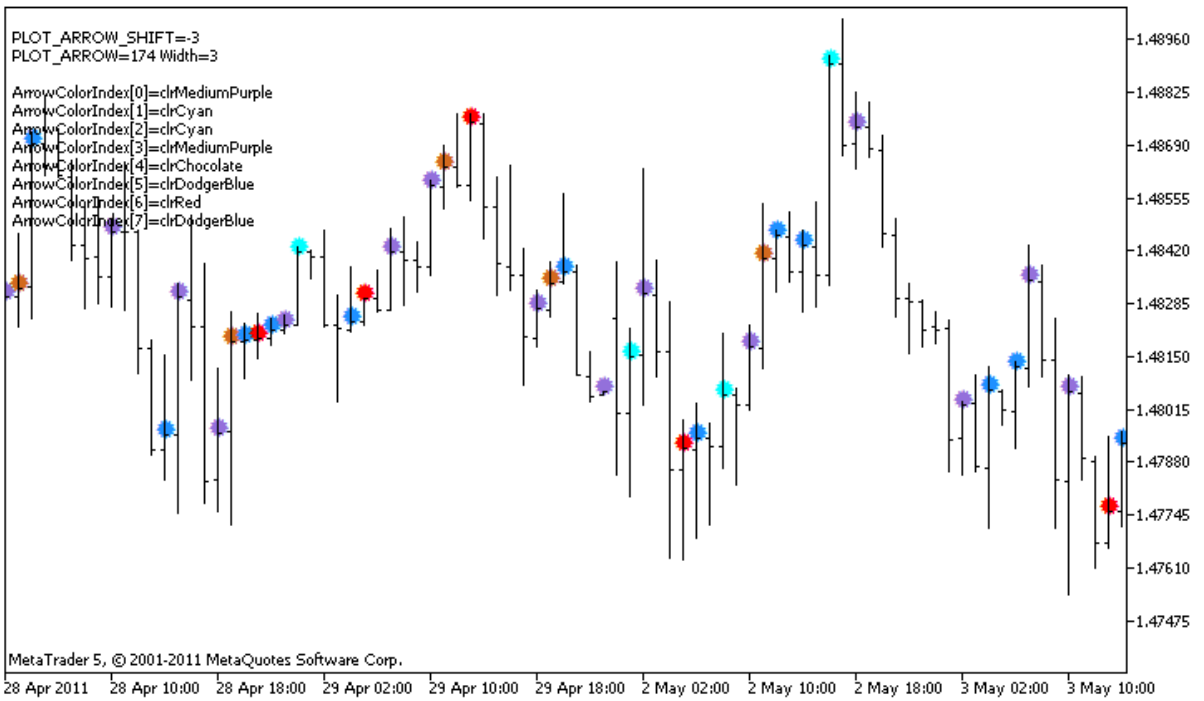
DRAW_COLOR_ARROW stili, bir alt pencerede veya çizelge ana penceresinde kullanılabilir. Boş değerler çizilmez ve "Veri Penceresinde" gösterilmezler; bu yüzden gösterge tamponlarındaki tüm değerler açıkça ayarlanmalıdır. Tamponlar sıfır değeri ile başlatılamazlar.

```
//--- Bir boş değer ayarla
   PlotIndexSetDouble(çizim_indisi_DRAW_COLOR_ARROW,PLOT_EMPTY_VALUE,0);
```

DRAW_COLOR_ARROW çizimi için gereken tamponların sayısı 2'dir.

- Sembol çiziminde kullanılacak fiyat değerini depolamak için bir tampon (ayrıca, PLOT_ARROW_SHIFT özelliğinde verilen piksel bazında bir kaydırma değeri);
- Sembol çiziminde kullanılacak renk indisini depolamak için bir tampon (sadece boş olmayan değerlerler anlamlıdır).

Bir önceki çubuğun kapanış fiyatından daha yüksek kapanış fiyatına sahip olan çubuğun üzerine ok çizen gösterge örneği **Tüm** okların genişlikleri, kaydırma değerleri ve sembol kodları, her **N** tikte bir değiştirilir. Sembolün rengi, üzerine çizildiği çubuğun numarasına bağlıdır.



DRAW_COLOR_ARROW stilindeki **plot1** örneğinde, renk ve boyut özellikleri, #property derleyici direktifi kullanılarak belirlenir, sonrasında iseOnCalculate() fonksiyonunda, özellikler rassal olarak ayarlanır. N parametresi, el ile yapılandırma olasılığı göz önüne alınarak, göstergenin dışsal parametreleri içinde ayarlanır (göstergenin özellikler penceresindeki Veriler sekmesi).

#property derleyici direktifi ile başlangıçta 8 renk ayarlandığını, daha sonra OnCalculate() fonksiyonu içerisinde, rengin colors[] dizisinde depolanan 14 renk arasından rassal olarak seçildiğini lütfen not ediniz.

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_ARROW.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_COLOR_ARROW stilini betimleyen bir gösterge"
#property description "Unicode karakterler ile, çizelge üzerinde farklı renklerde okla
#property description "Okun rengi, genişliği, kaydırma değeri ve sembol kodu, her N ti
#property description " rassal olarak değiştirilir"
#property description "code parametresi, temel değeri ayarlar: code=159 (daire)"

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   1
//--- plot ColorArrow
#property indicator_label1  "ColorArrow"
#property indicator_type1   DRAW_COLOR_ARROW
```

```
//--- Haftanın günleri temelinde, histogramın renklendirilmesi için 8 renk tanımla (öz
#property indicator_color1  clrRed,clrBlue,clrSeaGreen,clrGold,clrDarkOrange,clrMagent
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1

//--- giriş parametreleri
input int      N=5;          // Değiştirilecek tik sayısı
input ushort   code=159;     // DRAW_ARROW içinde çizilecek sembol kodu
int            color_sections;
//--- Grafik için bir gösterge tamponu
double         ColorArrowBuffer[];
//--- Renk indislerinin depolanması için bir dizi
double         ColorArrowColors[];
//--- Renkleri depolayacak 14 elemanlı bir dizi
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,ColorArrowBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorArrowColors,INDICATOR_COLOR_INDEX);
//--- PLOT_ARROW içinde çizilecek sembol kodunu tanımla
   PlotIndexSetInteger(0,PLOT_ARROW,code);
//--- Oklar için piksel bazında dikey kaydırma ayarla
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,5);
//--- Boş değer 0 olarak ayarla
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//---- Sinüsoidin renklendirilmesi için gereken renklerin sayısı
   color_sections=8;   //  #property indicator_color1 özelliğindeki yoruma bak
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- Rengin, boyutun, kaydırma değerinin ve sembol kodunun değiştirileceği tikleri he
   ticks++;
//--- Tikler önemli bir sayıda biriktirilmişse
   if(ticks>=N)
     {
      //--- Ok özelliklerini değiştir
      ChangeLineAppearance();
      //--- Histogramın çiziminde kullanılacak renkleri değiştir
      ChangeColors(colors,color_sections);
      //--- Tik sayacını sıfırla
      ticks=0;
     }

//--- Gösterge değerlerinin hesaplanması için bir blok
   int start=1;
   if(prev_calculated>0) start=prev_calculated-1;
//--- Hesap döngüsü
   for(int i=1;i<rates_total;i++)
     {
      //--- mevcut Close (kapanış) fiyatı öncekinden yüksekse, bir ok çiz
      if(close[i]>close[i-1])
         ColorArrowBuffer[i]=close[i];
      //--- Aksi durumda anlamsız değer belirt
      else
         ColorArrowBuffer[i]=0;
      //--- Ok rengi
      int index=i%color_sections;
      ColorArrowColors[i]=index;
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Çizgi kısımlarının renklerini değiştirir                         |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Renklerin sayısı
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Her renk indisi için rassal olarak bir renk tanımla
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
```

```
      //--- Rassal bir değer al
      int number=MathRand();
      //--- col[] dizisi içinde, tam-sayı bölümünden kalan şeklinde bir indisi al
      int i=number%size;
      //--- Her bir indis için, rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
      PlotIndexSetInteger(0,                    //  Bir grafiksel stilin numarası
                          PLOT_LINE_COLOR,      //  Özellik tanımlayıcı
                          plot_color_ind,       //  Rengin girildiği indis
                          cols[i]);             //  Yeni bir renk
      //--- Renkleri yaz
      comm=comm+StringFormat("ArrowColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStrin
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Göstergenin görüntülenen çizgisinin görünümünü değiştirir        |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Çizgi özelliklerine dair bilginin biçimlendirilmesi için bir dizgi
   string comm="";
//--- Çizgi kalınlığını değiştirmek için bir blok
   int number=MathRand();
//--- Tam-sayı bölümden kalan genişliği al
   int width=number%5; // Genişlik 0'dan 4'e ayarlanır
//--- Rengi PLOT_LINE_WIDTH özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Çizgi kalınlığını yaz
   comm=comm+" Width="+IntegerToString(width);

//--- Ok kodunu (PLOT_ARROW) değiştirmek için bir blok
   number=MathRand();
//--- Yeni ok kodunu hesaplamak için tam-sayı bölümünden kalanı al (0'dan 19'a)
   int code_add=number%20;
//--- code+code_add işleminin sonucu olarak yeni sembol kodunu ayarla
   PlotIndexSetInteger(0,PLOT_ARROW,code+code_add);
//--- PLOT_ARROW sembol kodunu yaz
   comm="\r\n"+"PLOT_ARROW="+IntegerToString(code+code_add)+comm;

//--- Okların dikey kaydırma değerini piksel bazında değiştirmek için bir blok
   number=MathRand();
//--- Kaydırma değerini tamsayı bölümünün kalanı olarak al
   int shift=20-number%41;
//--- Yeni kaydırma değerini şuradan ayarla
   PlotIndexSetInteger(0,PLOT_ARROW_SHIFT,shift);
//--- PLOT_ARROW_SHIFT kaydırma değerini yaz
   comm="\r\n"+"PLOT_ARROW_SHIFT="+IntegerToString(shift)+comm;
```

```
//--- Bir yorum kullanarak, çizelge üzerinde bilgi göster
   Comment(comm);
  }
```

DRAW_COLOR_ZIGZAG

DRAW_COLOR_ZIGZAG stili, iki gösterge tamponunun değerlerini kullanarak farklı renkte kısımlar (parçalar) çizer. Bu stil, DRAW_ZIGZAG stilinin renkli versiyonudur, yani her bir parça için ayrı bir rengin önceden ayarlanmış bir renk kümesi içerisinden seçilmesine izin verir. Parçalar ilk gösterge tamponunun değerinden ikinci tamponun değerine çizilir. Tamponların hiçbiri baştan sona boş değerler içeremez; böyle bir durumda hiçbir şey çizilmeyecektir.

Çizginin kalınlık, renk ve stilleri, DRAW_ZIGZAG stiline benzer şekilde, derleyici direktifleri ile veya dinamik olarak PlotIndexSetInteger() fonksiyonunun kullanımıyla belirtilebilir. Çizim özelliklerinin dinamik olarak değiştirilmesi, göstergelerin "canlılaştırılmasını" sağlar, bu şekilde göstergelerin görünümleri mevcut duruma göre değişir.

Parçalar, gösterge tamponunun boş olmayan bir değerinden, yine boş olmayan bir değerine kadar çizilir. Bir değerin "boş" olacağını belirtmek için, değeri PLOT_EMPTY_VALUE özelliğiyle ayarlayın:

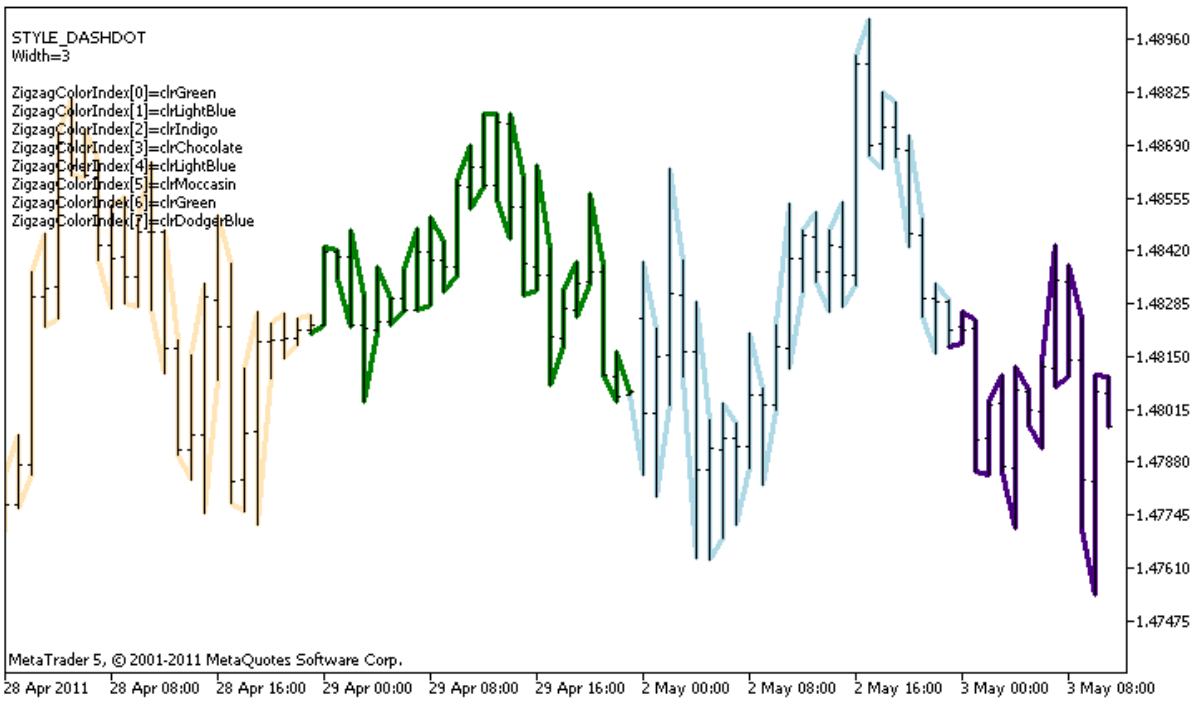
```
//--- 0 (boş) değer, çizime katılmayacak
   PlotIndexSetDouble(çizim_indisi_DRAW_COLOR_ZIGZAG,PLOT_EMPTY_VALUE,0);
```

Gösterge tamponlarının değerlerini her zaman açıkça doldurun, atlamak istediğiniz çubukları boş değer olarak ayarlayın.

DRAW_COLOR_ZIGZAG stilinin çizimi için gereken tampon sayısı 3'tür:

- iki tampon, zigzag parçalarının (kısımlarının) son değerlerini depolamak için;
- bir tampon, parçayı çizmekte kullanılan renk indisini depolamak için (sadece, boş olmayan değerlerler için ayarlanması mantıklıdır).

High ve Low fiyatların temelinde bir testere şekli çizen gösterge örneği. Parçaların renkleri, kalınlıkları ve stilleri **N** tikte bir rassal olarak değişir



DRAW_COLOR_ZIGZAG stiline sahip olan **plot1** için, #property derleyici direktifi ile başlangıçta 8 renk ayarlandığını, daha sonra OnCalculate() fonksiyonu içerisinde, rengin colors[] dizisinde depolanan 14 renk arasından rassal olarak seçildiğini lütfen not ediniz.

N parametresi, el ile yapılandırma olasılığı göz önüne alınarak, göstergenin dışsal parametreleri içinde ayarlanır (Gösterge Özellikleri penceresindeki Veriler sekmesi).

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_ZIGZAG.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_COLOR_ZIGZAG stilini betimleyen bir gösterge"
#property description "Renkli parçalar dizisi şeklinde bir kırık çizgi çizer, renk ise
#property description "Parçaların renk, genişlik ve stilleri, her N tikten sonra"
#property description " rassal olarak değiştirilir"

#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
//--- Kısımları renklendirmek için 8 renk tanımla (özel dizide depolanırlar)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- giriş parametreleri
input int      N=5;              // Değişim için gerekli tik sayısı
int            color_sections;
//--- Parça sonlarının değerleri için tamponlar
double         Color_ZigzagBuffer1[];
double         Color_ZigzagBuffer2[];
//--- Parça sonlarının renk indisleri için tamponlar
double         Color_ZigzagColors[];
//--- Renkleri depolayacak 14 elemanlı bir dizi
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrWhiteSmoke,clrCyan,clrMediumPurp
  };
//--- Çizgi stillerinin depolanması için bir dizi
ENUM_LINE_STYLE styles[]={STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
```

```
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,Color_ZigzagBuffer1,INDICATOR_DATA);
   SetIndexBuffer(1,Color_ZigzagBuffer2,INDICATOR_DATA);
   SetIndexBuffer(2,Color_ZigzagColors,INDICATOR_COLOR_INDEX);
//----Zigzag'ı renklendirmek için renk indisi
   color_sections=8;   //  #property indicator_color1 özelliğindeki yoruma bak
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- Çizginin stilini,rengini ve genişliğini değiştirmek için tikleri hesapla
   ticks++;
//--- Yeterli sayıda tik birikmişse
   if(ticks>=N)
     {
      //--- Çizgi özelliklerini değiştir
      ChangeLineAppearance();
      //--- Parçaların çiziminde kullanılan renkleri değiştir
      ChangeColors(colors,color_sections);
      //--- Tik sayacını sıfırla
      ticks=0;
     }

//--- Her bir çubuğun gün bilgisini almak için gereken zaman yapısı
   MqlDateTime dt;

//--- Hesaplamaların başlangıç pozisyonu
   int start=0;
//--- Eğer gösterge bir önceki tikte hesaplanmışsa, hesaplamayı sondan bir önceki çubu
   if(prev_calculated!=0) start=prev_calculated-1;
//--- Hesap döngüsü
   for(int i=start;i<rates_total;i++)
```

```
      {
       //--- Yapı içinde çubuk açılış zamanını yaz
       TimeToStruct(time[i],dt);

       //--- Çubuk numarası çift ise
       if(i%2==0)
         {
          //---  High değerini 1-inci tampona ve Low değerini 2-inci tampona yaz one
          Color_ZigzagBuffer1[i]=high[i];
          Color_ZigzagBuffer2[i]=low[i];
          //--- Parçanın rengi
          Color_ZigzagColors[i]=dt.day_of_year%color_sections;
         }
       //--- çubuk numarası tek ise
       else
         {
          //--- Çubukları ters sırada doldur
          Color_ZigzagBuffer1[i]=low[i];
          Color_ZigzagBuffer2[i]=high[i];
          //--- Parçanın rengi
          Color_ZigzagColors[i]=dt.day_of_year%color_sections;
         }
      }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Zigzag kısımların rengini değiştirir                             |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Renklerin sayısı
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Her renk indisi için rassal olarak bir renk tanımla
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Rassal bir değer al
      int number=MathRand();
      //--- col[] dizisi içinde, tam-sayı bölümünden kalan şeklinde bir indisi al
      int i=number%size;
      //--- Her bir indis için, rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
      PlotIndexSetInteger(0,                    //  Bir grafiksel stilin numarası
                          PLOT_LINE_COLOR,      //  Özellik tanımlayıcı
                          plot_color_ind,       //  Rengin girildiği indis
                          cols[i]);             //  Yeni bir renk
      //--- Renkleri yaz
```

```
      comm=comm+StringFormat("ZigzagColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Zigzag parçalarının görünümünü değiştirir                        |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Color_ZigZag özelliklerine dair bilginin biçimlendirilmesi için bir dizgi
   string comm="";
//--- Çizgi kalınlığını değiştirmek için bir blok
   int number=MathRand();
//--- Tam-sayı bölümden kalan genişliği al
   int width=number%5;   // Genişlik 0'dan 4'e ayarlandı
//--- Rengi PLOT_LINE_WIDTH özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Çizgi kalınlığını yaz
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Çizgi stilini değiştirmek için bir blok
   number=MathRand();
//--- Bölen, syles dizisinin büyüklüğüne eşit
   int size=ArraySize(styles);
//--- Yeni bir stil seçmek için, indis değerini, bölümden kalan tam-sayı değeri şeklir
   int style_index=number%size;
//--- Rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_STYLE,styles[style_index]);
//--- Çizgi stilini yaz
   comm="\r\n"+EnumToString(styles[style_index])+""+comm;
//--- Bir yorum kullanarak, çizelge üzerinde bilgi göster
   Comment(comm);
  }
```

DRAW_COLOR_BARS

DRAW_COLOR_BARS stili Open, High, Low ve Close fiyatlarını içeren dört gösterge tamponunun temelinde çubuklar çizer. Bu stil, DRAW_BARS stilinin gelişmiş bir versiyonudur ve önceden tanımlanmış bir renk kümesi içinden, her çubuk için ayrı bir rengin belirlenmesini sağlar. Bu stil, bir çizelge alt penceresinde yer alanlar ve diğer finansal araçlar da dahil olmak üzere, çubuklar şeklinde göstergeler oluşturulması için tasarlanmıştır.

Çubukların renkleri derleyici direktifleri ile veya dinamik olarak, PlotIndexSetInteger() fonksiyonunu kullanarak ayarlanabilir. Çizim özelliklerinin dinamik olarak değiştirilmesi, göstergelerin "canlılaştırılmasını" sağlar, bu şekilde göstergelerin görünümleri mevcut duruma göre değişir.

Gösterge, dört gösterge tamponundan **hiçbirinin** boş değer içermediği çubuklar için çizilir. Bir değerin "boş" olarak göz önüne alınacağını belirtmek için, bu değeri PLOT_EMPTY_VALUE özelliğine ayarlayın:

```
//--- 0 (boş) değer, çizime katılmayacak
   PlotIndexSetDouble(çizim_indisi_DRAW_COLOR_BARS,PLOT_EMPTY_VALUE,0);
```

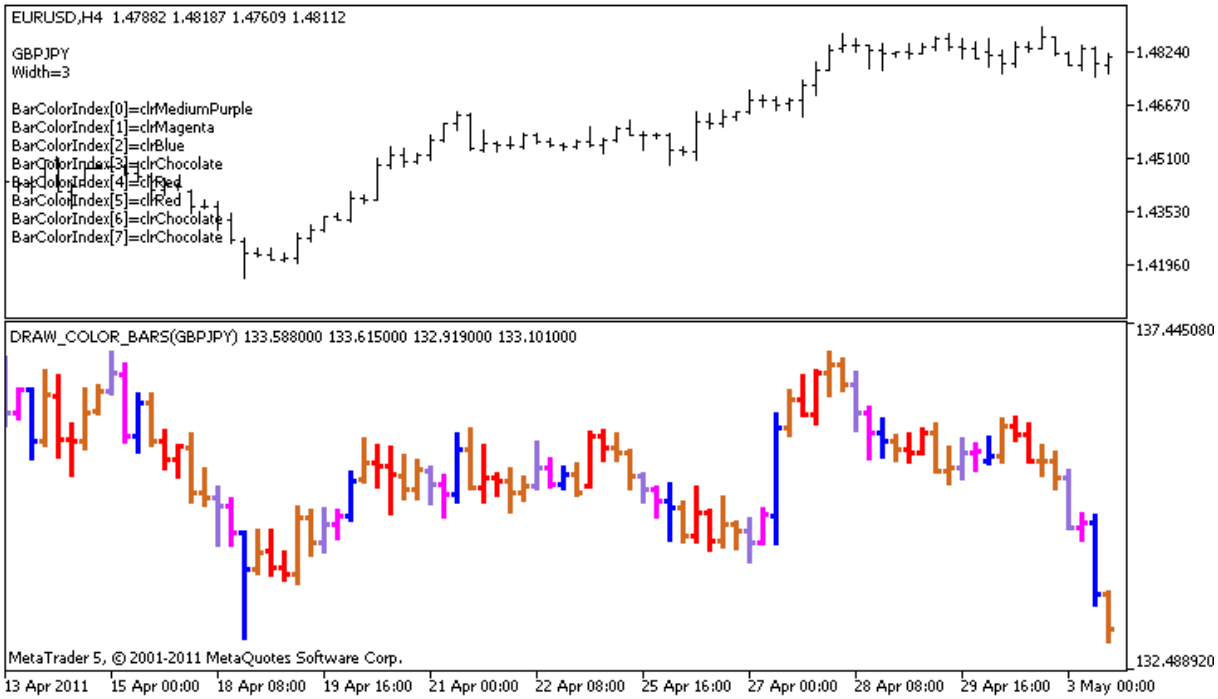
Gösterge tamponlarının değerlerini her zaman açıkça doldurun, atlamak istediğiniz çubukları boş değer olarak ayarlayın.

DRAW_COLOR_BARS stilinin çizimi için gereken tampon sayısı 5'tir:

- Open, High, Low ve Close değerleri için dört tampon;
- çubuğa çizilecek rengin indisini depolamak için bir tampon (ayarlanması, sadece çizilecek çubuklar için anlam taşır).

Çizilecek tüm tamponlar birbiri ardına, verilen sırayla dizilmelidir: Open, High, Low, Close ve renk tamponu. Fiyat tamponlarının hiçbiri baştan sona boş değerler içeremez; böyle bir durumda hiçbir şey çizilmeyecektir.

Ayrı bir pencerede yer alan seçilmiş finansal aracın üzerine çubuklar çizen gösterge örneği. Çubukların rengi her **N** tikte rassal olarak değişir. N parametresi, el ile yapılandırma olasılığı göz önüne alınarak, göstergenin dışsal parametreleri içinde ayarlanır (göstergenin Özellikler penceresindeki Veriler sekmesi).



DRAW_COLOR_BARS stilindeki **plot1** için, #property derleyici direktifi ile başlangıçta 8 renk ayarlandığını, daha sonra OnCalculate() fonksiyonu içerisinde, rengin colors[] dizisinde depolanan 14 renk arasından rassal olarak seçildiğini lütfen not ediniz.

```
//+------------------------------------------------------------------+
//|                                                DRAW_COLOR_BARS.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_COLOR_BARS stilini betimleyen bir gösterge"
#property description "Ayrı bir penceredeki, seçilen sembolün çubuklarını farklı renkl
#property description "Çubukların renk ve genişliği ve ayrıca sembol, her N tikten sor
#property description "rassal olarak değiştirilir"

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plot ColorBars
#property indicator_label1  "ColorBars"
#property indicator_type1   DRAW_COLOR_BARS
//--- Çubukları renklendirmek için 8 renk tanımla (özel dizide depolanırlar)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrL
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- giriş parametreleri
input int      N=5;              // Tipin değişmesi için gereken tik sayısı
```

```
input int      bars=500;         // Gösterilecek çubuk sayısı
input bool     messages=false;   // "Uzmanlar" günlüğünde mesajları göster
//--- Gösterge tamponları
double         ColorBarsBuffer1[];
double         ColorBarsBuffer2[];
double         ColorBarsBuffer3[];
double         ColorBarsBuffer4[];
double         ColorBarsColors[];
//--- Sembol ismi
string symbol;
int    bars_colors;
//--- Renkleri depolayacak 14 elemanlı bir dizi
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
  };
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   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);
//---- Çubukların renklendirilmesi için kullanılacak renklerin sayısı
   bars_colors=8;   //  #property indicator_color1 özelliğindeki yoruma bak
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- Çubuğun renginin, genişliğinin ve stilinin değiştirileceği tikleri say
```

```
   ticks++;
//--- Yeterli sayıda tik birikmişse
   if(ticks>=N)
     {
      //--- Piyasa Gözlemi penceresinden yeni bir sembol seç
      symbol=GetRandomSymbolName();
      //--- Çizgi özelliklerini değiştir
      ChangeLineAppearance();
      //--- Mumların çizimi için kullanılan renkleri değiştir
      ChangeColors(colors,bars_colors);
      int tries=0;
      //--- Tamponları, sembolün fiyatlarıyla doldurmak için 5 deneme gerçekleştir
      while(!CopyFromSymbolToBuffers(symbol,rates_total,bars_colors) && tries<5)
        {
         //--- CopyFromSymbolToBuffers() fonksiyonunun çağrıları için bir sayaç
         tries++;
        }
      //--- Tik sayacını sıfırla
      ticks=0;
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Gösterge tamponlarını fiyatlarla doldur                          |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,int total,int bar_colors)
  {
//--- rates[] dizisinde, Open, High, Low ve Close fiyatlarını kopyalayacağız
   MqlRates rates[];
//--- Deneme sayacı
   int attempts=0;
//--- Kaç tane kopyalandı
   int copied=0;
//--- Istenen sembole dair bir zaman serisi elde etmek için 25 deneme gerçekleştir
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) denemeler=%d",__FUNCTION__,name,atter
     }
//--- Yeterli sayıda çubuk kopyalanamamışsa
   if(copied!=bars)
     {
      //--- Bir mesaj dizgisinden
      string comm=StringFormat("%s sembolü için, sadece %d çubuk alınabildi (istenen %
                               name,
                               copied,
                               bars
```

```
                                  );
      //--- Mesajı, ana çizelge penceresindeki bir yorumda göster
      Comment(comm);
      //--- Mesajı göster
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- Sembolün görüntüsünü ayarla
      PlotIndexSetString(0,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;"+name+"
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_BARS("+name+")");
     }
//--- Tamponları boş değerlerle başlat
   ArrayInitialize(ColorBarsBuffer1,0.0);
   ArrayInitialize(ColorBarsBuffer2,0.0);
   ArrayInitialize(ColorBarsBuffer3,0.0);
   ArrayInitialize(ColorBarsBuffer4,0.0);

//--- Fiyatları tamponlara kopyala
   for(int i=0;i<copied;i++)
     {
      //--- Tamponlar için uygun indisleri hesapla
      int buffer_index=total-copied+i;
      //--- Fiyatları tamponlara yaz
      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);
  }
//+------------------------------------------------------------------+
//| Piyasa Gözleminden rassal bir sembole dönüş yapar                |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- Piyasa Gözleminde görünen sembollerin sayısı
   int symbols=SymbolsTotal(true);
//--- Sembolün listedeki pozisyonu - 0'dan symbols'e kadar rassal bir sayı
   int number=MathRand()%symbols;
//--- Belirtilen konumdaki sembolün ismine dönüş yap
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Zigzag kısımların rengini değiştirir                             |
//+------------------------------------------------------------------+
```

```
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Renklerin sayısı
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Her renk indisi için rassal olarak bir renk tanımla
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Rassal bir değer al
      int number=MathRand();
      //--- col[] dizisi içinde, tam-sayı bölümünden kalan şeklinde bir indisi al
      int i=number%size;
      //--- Her bir indis için, rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
      PlotIndexSetInteger(0,                    //  Bir grafiksel stilin numarası
                          PLOT_LINE_COLOR,      //  Özellik tanımlayıcı
                          plot_color_ind,       //  Rengin girildiği indis
                          cols[i]);             //  Yeni bir renk
      //--- Renkleri yaz
      comm=comm+StringFormat("BarColorIndex[%d]=%s \r\n",plot_color_ind,ColorToString
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Çubukların görünümünü değiştirir                                 |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Çubuk özellikleri ile ilgili bilgiyi biçimlendirmek için bir dizgi
   string comm="";

//--- Çubuk genişliğini değiştirmek için bir blok
   int number=MathRand();
//--- Tam-sayı bölümden kalan genişliği al
   int width=number%5;   // Genişlik 0'dan 4'e ayarlandı
//--- Rengi PLOT_LINE_WIDTH özelliği şeklinde ayarla
   PlotIndexSetInteger(0,PLOT_LINE_WIDTH,width);
//--- Çizgi kalınlığını yaz
   comm=comm+"\r\nWidth="+IntegerToString(width);

//--- Sembol ismini yaz
   comm="\r\n"+symbol+comm;

//--- Bir yorum kullanarak, çizelge üzerinde bilgi göster
   Comment(comm);
  }
```

DRAW_COLOR_CANDLES

The DRAW_COLOR_CANDLES stili, DRAW_CANDLES stili gibi; Open, High, Low ve Close fiyatlarını içeren dört gösterge tamponunun değerlerini kullanarak mumlar çizer. Ayrıca, her bir mumun renginin verilen bir küme içerisinden seçilmesine olanak tanır. Bu amaçla stil, her bir çubuk için kullanılacak rengin indisini depolayan özel bir tampon içerir. Ayrı bir çizelge alt penceresinde yer alanlar ve diğer finansal araçlar da dahil olmak üzere, mum şeklinde göstergeler oluşturulması için tasarlanmıştır.

Mum renklerinin sayısı, derleyici direktifleri ile veya dinamik olarak PlotIndexSetInteger() fonksiyonunu kullanarak ayarlanabilir. Çizim özelliklerinin dinamik olarak değiştirilmesi, göstergelerin "canlılaştırılmasını" sağlar, bu şekilde göstergelerin görünümleri mevcut duruma göre değişir.

Gösterge, dört fiyat tamponundan hiçbirinin boş değer içermediği çubuklar için ayarlanır. Bir değerin "boş" olacağını belirtmek için, değeri PLOT_EMPTY_VALUE özelliğiyle ayarlayın:

```
//--- 0 (boş) değer, çizime katılmayacak
   PlotIndexSetDouble(çizim_indisi_DRAW_COLOR_CANDLES,PLOT_EMPTY_VALUE,0);
```

Gösterge tamponlarının değerlerini her zaman açıkça doldurun, atlamak istediğiniz çubukları boş değer olarak ayarlayın.

DRAW_COLOR_CANDLES stilinin çizimi için gereken tamponların sayısı 5'tir:

- Open, High, Low ve Close değerleri için dört tampon;
- çubuğa çizilecek rengin indisini depolamak için bir tampon (ayarlanması, sadece çizilecek çubuklar için anlam taşır).

Çizilecek tüm tamponlar birbiri ardına, verilen sırayla dizilmelidir: Open, High, Low, Close ve renk tamponu. Fiyat tamponlarının hiçbiri baştan sona boş değerler içeremez; böyle bir durumda hiçbir şey çizilmeyecektir.

Ayrı bir pencerede yer alan seçilmiş finansal aracın üzerine mumlar çizen gösterge örneği. Mumların rengi her **N** tikte rassal olarak değişir. N parametresi, el ile yapılandırma olasılığı göz önüne alınarak, göstergenin dışsal parametreleri içinde ayarlanır (göstergenin Özellikler penceresindeki Veriler sekmesi).

plot1 için, renk ve boyut özelliklerinin #property derleyici direktifi kullanılarak belirlendiğini; sonrasında ise rengin, OnCalculate() fonksiyonu içinde önceden hazırlanmış bir listeden rassal olarak ayarlandığını lütfen not ediniz.

```
//+------------------------------------------------------------------+
//|                                              DRAW_COLOR_CANDLES.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"

#property description "DRAW_COLOR_CANDLES stilini betimlemek için bir gösterge."
#property description "Seçilen sembolün mumlarını ayrı bir pencerede çizer"
#property description " "
#property description "Çubukların renk ve genişliği ve ayrıca sembol, her N tikten sor
#property description "rassal olarak değiştirilir"

#property indicator_separate_window
#property indicator_buffers 5
#property indicator_plots   1
//--- plot ColorCandles
#property indicator_label1  "ColorCandles"
#property indicator_type1   DRAW_COLOR_CANDLES
//--- Mumları renklendirmek için 8 renk tanımla (özel dizide depolanırlar)
#property indicator_color1  clrRed,clrBlue,clrGreen,clrYellow,clrMagenta,clrCyan,clrLi
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

```
//--- giriş parametreleri
input int      N=5;              // Tipin değişmesi için gereken tik sayısı
input int      bars=500;         // Gösterilecek mumların sayısı
input bool     messages=false;   // "Uzmanlar" günlüğünde mesajları göster
//--- Gösterge tamponları
double         ColorCandlesBuffer1[];
double         ColorCandlesBuffer2[];
double         ColorCandlesBuffer3[];
double         ColorCandlesBuffer4[];
double         ColorCandlesColors[];
int            candles_colors;
//--- Sembol ismi
string symbol;
//--- Renkleri depolayacak 14 elemanlı bir dizi
color colors[]=
  {
   clrRed,clrBlue,clrGreen,clrChocolate,clrMagenta,clrDodgerBlue,clrGoldenrod,
   clrIndigo,clrLightBlue,clrAliceBlue,clrMoccasin,clrMagenta,clrCyan,clrMediumPurple
  };
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Çubuklar çok küçükse - işi planlanandan önce bitir
   if(bars<50)
     {
      Comment("Lüyfen daha fazla sayıda çubuk belirtin! Gösterge işlemi sonlandırıldı"
      return(INIT_PARAMETERS_INCORRECT);
     }
//--- gösterge tamponlarının eşlenmesi
   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);
//--- Bir boş değer
   PlotIndexSetDouble(0,PLOT_EMPTY_VALUE,0);
//--- Çubukların çizileceği sembolün ismi
   symbol=_Symbol;
//--- Sembolün görünümünü ayarla
   PlotIndexSetString(0,PLOT_LABEL,symbol+" Open;"+symbol+" High;"+symbol+" Low;"+symb
   IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
//---- Mumların renklendirilmesi için gereken renklerin sayısı
   candles_colors=8;     //   #property indicator_color1 özelliğindeki yoruma bakınız
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

```
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- Stili ve rengi değiştirmek için tikleri say
   ticks++;
//--- Yeterli sayıda tik birikmişse
   if(ticks>=N)
     {
      //--- Piyasa Gözlemi penceresinden yeni bir sembol seç
      symbol=GetRandomSymbolName();
      //--- Biçimi değiştir
      ChangeLineAppearance();
      //--- Mumların çizimi için kullanılan renkleri değiştir
      ChangeColors(colors,candles_colors);

      int tries=0;
      //--- plot1 tamponlarını, sembolün fiyatlarıyla doldurmak için 5 deneme yap
      while(!CopyFromSymbolToBuffers(symbol,rates_total,0,
            ColorCandlesBuffer1,ColorCandlesBuffer2,ColorCandlesBuffer3,
            ColorCandlesBuffer4,ColorCandlesColors,candles_colors)
            && tries<5)
        {
         //--- CopyFromSymbolToBuffers() fonksiyonunun çağrıları için bir sayaç
         tries++;
        }
      //--- Tik sayacını sıfırla
      ticks=0;
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Belirtilen mumu doldurur                                         |
//+------------------------------------------------------------------+
bool CopyFromSymbolToBuffers(string name,
                             int total,
                             int plot_index,
                             double &buff1[],
```

```
                              double &buff2[],
                              double &buff3[],
                              double &buff4[],
                              double &col_buffer[],
                              int    cndl_colors
                              )
  {
//--- rates[] dizisinde, Open, High, Low ve Close fiyatlarını kopyalayacağız
   MqlRates rates[];
//--- Deneme sayacı
   int attempts=0;
//--- Kaç tane kopyalandı
   int copied=0;
//--- Istenen sembole dair bir zaman serisi elde etmek için 25 deneme gerçekleştir
   while(attempts<25 && (copied=CopyRates(name,_Period,0,bars,rates))<0)
     {
      Sleep(100);
      attempts++;
      if(messages) PrintFormat("%s CopyRates(%s) denemeler=%d",__FUNCTION__,name,atter
     }
//--- Yeterli sayıda çubuk kopyalanamamışsa
   if(copied!=bars)
     {
      //--- Bir mesaj dizgisinden
      string comm=StringFormat("%s sembolü için, sadece %d çubuk alınabildi (istenen %
                               name,
                               copied,
                               bars
                               );
      //--- Mesajı, ana çizelge penceresindeki bir yorumda göster
      Comment(comm);
      //--- Mesajı göster
      if(messages) Print(comm);
      return(false);
     }
   else
     {
      //--- Sembolün görüntüsünü ayarla
      PlotIndexSetString(plot_index,PLOT_LABEL,name+" Open;"+name+" High;"+name+" Low;
      IndicatorSetString(INDICATOR_SHORTNAME,"DRAW_COLOR_CANDLES("+symbol+")");
     }
//--- Tamponları boş değerlerle başlat
   ArrayInitialize(buff1,0.0);
   ArrayInitialize(buff2,0.0);
   ArrayInitialize(buff3,0.0);
   ArrayInitialize(buff4,0.0);
//--- Her tik ile birlikte fiyatları tamponlara kopyala
   for(int i=0;i<copied;i++)
     {
```

```
      //--- Tamponlar için uygun indisleri hesapla
      int buffer_index=total-copied+i;
      //--- Fiyatları tamponlara yaz
      buff1[buffer_index]=rates[i].open;
      buff2[buffer_index]=rates[i].high;
      buff3[buffer_index]=rates[i].low;
      buff4[buffer_index]=rates[i].close;
      //--- Mum rengini ayarla
      int color_index=i%cndl_colors;
      col_buffer[buffer_index]=color_index;
     }
   return(true);
  }
//+------------------------------------------------------------------+
//| Piyasa Gözleminden rassal bir sembole dönüş yapar                |
//+------------------------------------------------------------------+
string GetRandomSymbolName()
  {
//--- Piyasa Gözleminde görünen sembollerin sayısı
   int symbols=SymbolsTotal(true);
//--- Sembolün listedeki pozisyonu - 0'dan symbols'e kadar rassal bir sayı
   int number=MathRand()%symbols;
//--- Belirtilen konumdaki sembolün ismine dönüş yap
   return SymbolName(number,true);
  }
//+------------------------------------------------------------------+
//| Mum kısımlarının renklerini değiştirir                           |
//+------------------------------------------------------------------+
void  ChangeColors(color  &cols[],int plot_colors)
  {
//--- Renklerin sayısı
   int size=ArraySize(cols);
//---
   string comm=ChartGetString(0,CHART_COMMENT)+"\r\n\r\n";

//--- Her renk indisi için rassal olarak bir renk tanımla
   for(int plot_color_ind=0;plot_color_ind<plot_colors;plot_color_ind++)
     {
      //--- Rassal bir değer al
      int number=MathRand();
      //--- col[] dizisi içinde, tam-sayı bölümünden kalan şeklinde bir indisi al
      int i=number%size;
      //--- Her bir indis için, rengi PLOT_LINE_COLOR özelliği şeklinde ayarla
      PlotIndexSetInteger(0,                    //  Bir grafiksel stilin numarası
                          PLOT_LINE_COLOR,      //  Özellik tanımlayıcı
                          plot_color_ind,       //  Rengin girildiği indis
                          cols[i]);             //  Yeni bir renk
      //--- Renkleri yaz
      comm=comm+StringFormat("CandleColorIndex[%d]=%s \r\n",plot_color_ind,ColorToStri
```

```
      ChartSetString(0,CHART_COMMENT,comm);
     }
//---
  }
//+------------------------------------------------------------------+
//| Mumların görünümünü değiştirir                                   |
//+------------------------------------------------------------------+
void ChangeLineAppearance()
  {
//--- Mum özelliklerine dair bilginin biçimlendirilmesi için bir dizgi
   string comm="";
//--- Sembol ismini yaz
   comm="\r\n"+symbol+comm;
//--- Bir yorum kullanarak, çizelge üzerinde bilgi göster
   Comment(comm);
  }
```

Gösterge Özellikleri ve Karşılık Gelen Fonksiyonlar Arasındaki Bağlantı

Her özel gösterge belli sayıda özelliğe sahiptir, bunlardan bazıları zorunludur ve her zaman açıklamanın başında konumlandırılırlar. Bunlar şu özellikleridir:

- göstergenin çizileceği pencerenin belirtilmesi - indicator_separate_window veya indicator_chart_window;
- gösterge tamponlarının sayısı - indicator_buffers;
- gösterge grafiklerinin sayısı - indicator_plots.

Bir de, hem önişlemci direktifleri, hem de özel göstergelerin oluşturulması için tasarlanmış fonksiyonları kullanarak ayarlanabilen özellikler mevcuttur. Bu özellikler ve karşılık gelen fonksiyonlar aşağıdaki tabloda tanımlanmıştır.

Gösterge alt-penceresi için direktifler	IndicatorSet...() tipi fonksiyonlar	Ayarlanmış alt-pencere özelliğinin açıklaması
indicator_height	IndicatorSetInteger (INDICATOR_INDICATOR_HEIGHT, nHeight)	Alt-pencere yüksekliğinin sabit değeri
indicator_minimum	IndicatorSetDouble (INDICATOR_MINIMUM, dMaxValue)	Dikey eksenin minimal değeri
indicator_maximum	IndicatorSetDouble (INDICATOR_MAXIMUM, dMinValue)	Dikey eksenin maksimal değeri
indicator_levelN	IndicatorSetDouble (INDICATOR_LEVELVALUE, N-1, nLevelValue)	N seviyesi için dikey eksen değeri
önişlemci direktifi yok	IndicatorSetString (INDICATOR_LEVELTEXT, N-1, sLevelName)	Görüntülenen seviyenin ismi
indicator_levelcolor	IndicatorSetInteger (INDICATOR_LEVELCOLOR, N-1, nLevelColor)	N seviyesinin rengi
indicator_levelwidth	IndicatorSetInteger (INDICATOR_LEVELWIDTH, N-1, nLevelWidth)	N seviyesinin kalınlığı
indicator_levelstyle	IndicatorSetInteger (INDICATOR_LEVELSTYLE, N-1, nLevelStyle)	N seviyesinin çizgi stili
Çizim özellikleri için direktifler	**PlotIndexSet...() tipi fonksiyonlar**	**Ayarlanan bir grafik özelliğinin açıklaması**

Gösterge alt-penceresi için direktifler	IndicatorSet...() tipi fonksiyonlar	Ayarlanmış alt-pencere özelliğinin açıklaması
indicator_labelN	PlotIndexSetString(N-1,PLOT_LABEL,sLabel)	N numaralı grafiğin kısa ismi. Veri Penceresinde ve - fare ile üzerine gelindiğinde - açılır araç-ipucu kutusu içinde görüntülenir
indicator_colorN	PlotIndexSetInteger(N-1, PLOT_LINE_COLOR, nColor)	N numaralı grafiğin çizgi rengi
indicator_styleN	PlotIndexSetInteger(N-1, PLOT_LINE_STYLE, nType)	N numaralı grafiğin çizgi stili
indicator_typeN	PlotIndexSetInteger(N-1, PLOT_DRAW_TYPE, nType)	N numaralı grafiğin çizgi tipi
indicator_widthN	PlotIndexSetInteger(N-1, PLOT_LINE_WIDTH, nWidth)	N numaralı grafiğin çizgi kalınlığı
Genel gösterge özellikleri	**IndicatorSet...() tipi fonksiyonlar**	**Açıklama**
önişlemci direktifi yok	IndicatorSetString (INDICATOR_SHORTNAME, sShortName)	Gösterge için, gösterge listesinde görüntülenecek uygun bir kısa isim ayarlar (**Ctrl+I** tuşlarına basıldığında terminalde açılır).
önişlemci direktifi yok	IndicatorSetInteger (INDICATOR_DIGITS, nDigits)	Gösterge değerleri için görüntülenecek kesinliği ayarlar - ondalık hane sayısı
önişlemci direktifi yok	IndicatorSetInteger (INDICATOR_LEVELS, nLevels)	Gösterge penceresindeki seviye sayısını ayarlar
indicator_applied_price	Fonksiyon yoktur, özellik sadece önişlemci direktifi ile ayarlanabilir.	Gösterge hesabı için varsayılan fiyat tipi kullanılır. Sadece gerekli olduğunda; OnCalculate() çağrısının birinci tipi kullanılıyorsa belirtilir. Bu özellik aynı zamanda göstergenin özellikler iletişim kutusunun "Veriler" sekmesinde de ayarlanabilir - "Uygula".

Ayrıca not edilmelidir ki, seviyelerin numaralandırılması, önişlemci içinde bir ile başlarken, aynı özelliklerin fonksiyonlarla kullanılması durumunda sıfır ile başlar; belirtilen değer, #property için belirtilenden 1 eksiktir.

Karşılık gelen bir fonksiyonun bulunmadığı birkaç direktif bulunmaktadır:

Direktif	Açıklama
indicator_chart_window	Gösterge ana çizelge penceresinde görüntülenir
indicator_separate_window	Gösterge ayrı bir alt-pencerede görüntülenir
indicator_buffers	Gereken gösterge tamponlarının sayısı
indicator_plots	Göstergedeki grafiklerin sayısı

SetIndexBuffer

Bu fonksiyon, belirtilen bir gösterge tamponunu double tipi dinamik bir dizi ile bağlar.

```
bool  SetIndexBuffer(
   int                   index,          // tampon indisi
   double                buffer[],       // dizi
   ENUM_INDEXBUFFER_TYPE  data_type      // depolanacak verinin tipi
   );
```

Parametreler

index

[in] Gösterge tamponunun numarası. Numaralama (indisleme) 0 ile başlar. İndis #property indicator_buffers özelliğinde bildirilen sayıdan küçük olmalıdır.

buffer[]

[in] Özel gösterge programında bildirilmiş bir dizi.

data_type

[in] Gösterge dizisinde depolanan verinin tipi. Varsayılan olarak INDICATOR_DATA değerini alır (hesaplanan göstergenin değerlerini belirtir). INDICATOR_COLOR_INDEX değerini de alabilir; bu durumda tampon, önceki gösterge tamponu için renk indislerini depolamak amacıyla kullanılır. 64 adete kadar farklı rengi #property indicator_colorN satırında belirtebilirsiniz. INDICATOR_CALCULATIONS değeri, tamponun ara hesaplamalarda kullanılmak için tasarlandığı ve çizilmeyeceği anlamına gelir.

Dönüş değeri

Başarılı sonuç durumunda true değerine, aksi durumda - false değerine dönüş yapar.

Not

İndisleme yönü daha önceden zaman-serileri şeklinde ayarlanmış bile olsa, buffer[] dizisi, bağlama işleminden sonra sıradan dizilerde olduğu gibi indislenir. Bir gösterge dizisindeki elemanların erişim sırasını değiştirmek istiyorsanız, ArraySetAsSeries() fonksiyonunu kullanın (**dizi SetIndexBuffer() fonksiyonu ile bağlandıktan sonra**). SetIndexBuffer() fonksiyonunu kullanarak gösterge tamponu şeklinde ayarlanmış dinamik bir dizinin büyüklüğünün değiştirilemeyeceğini lütfen not ediniz. Gösterge tamponlarının büyüklüklerinin değişimi için gerekli tüm işlemler, terminalin idari alt-sistemi tarafından gerçekleştirilir.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                TestCopyBuffer1.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                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
//--- giriş parametreleri
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;
//--- gösterge tamponları
double                   MABuffer[];
int                      ma_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   if(AsSeries) ArraySetAsSeries(MABuffer,true);
   Print("SetIndexBuffer() çağrısından sonra gösterge tamponu bir zaman-serisi = ",
   SetIndexBuffer(0,MABuffer,INDICATOR_DATA);
   Print("SetIndexBuffer() çağrısından sonra gösterge tamponu bir zaman-serisi = ",
         ArrayGetAsSeries(MABuffer));

//--- gösterge tamponu elemanlarının erişim sırasını değiştir
   ArraySetAsSeries(MABuffer,AsSeries);

   IndicatorSetString(INDICATOR_SHORTNAME,"MA("+period+")"+AsSeries);
//---
   ma_handle=iMA(Symbol(),0,period,shift,smootMode,price);
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- MABuffer tamponundaki hareketli ortalama değerlerini kopyala
   int copied=CopyBuffer(ma_handle,0,0,rates_total,MABuffer);

   Print("MABuffer[0] = ",MABuffer[0]);// AsSeries değerine bağlı olarak
                                       // Ya çok eski bir değer alınacak
                                       // Ya da tamamlanmamış bir çubuk

//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

Ayrıca Bakınız

Özel Gösterge Özellikleri, Zaman-serilerine ve göstergelere erişim

IndicatorSetDouble

Karşılık gelen gösterge özelliğinin değerini ayarlar. Gösterge özelliği double tipinde olmalıdır. Fonksiyonun iki çeşidi bulunmaktadır.

Özellik tanımlayıcısı (kimliği) ile çağrı.

```
bool  IndicatorSetDouble(
   int     prop_id,           // tanımlayıcı
   double  prop_value         // ayarlanacak değer
   );
```

Özellik tanımlayıcısının ve şekillendiricinin belirtilmesi ile yapılan çağrı.

```
bool  IndicatorSetDouble(
   int     prop_id,           // tanımlayıcı
   int     prop_modifier,     // şekillendirici
   double  prop_value         // ayarlanacak değer
   )
```

Parametreler

prop_id

[in] Gösterge özelliğinin tanımlayıcısı. Bu değer, ENUM_CUSTOMIND_PROPERTY_DOUBLE sayımının değerlerinden biri olabilir.

prop_modifier

[in] Belirtilen özelliğin şekillendiricisi. Sadece seviye özellikleri bir şekillendirici gerektirir. seviyelerin indislenmesi 0'dan başlar. Yani, ikinci seviyenin özelliğinin ayarlanması için '1' şekillendiricisini belirtmelisiniz (derleyici direktifleri ile yapılan kullanımdan 1 daha az).

prop_value

[in] Özellik değeri.

Dönüş değeri

Başarılı sonuç durumunda 'true', aksi durumda 'false' değerine dönüş yapar.

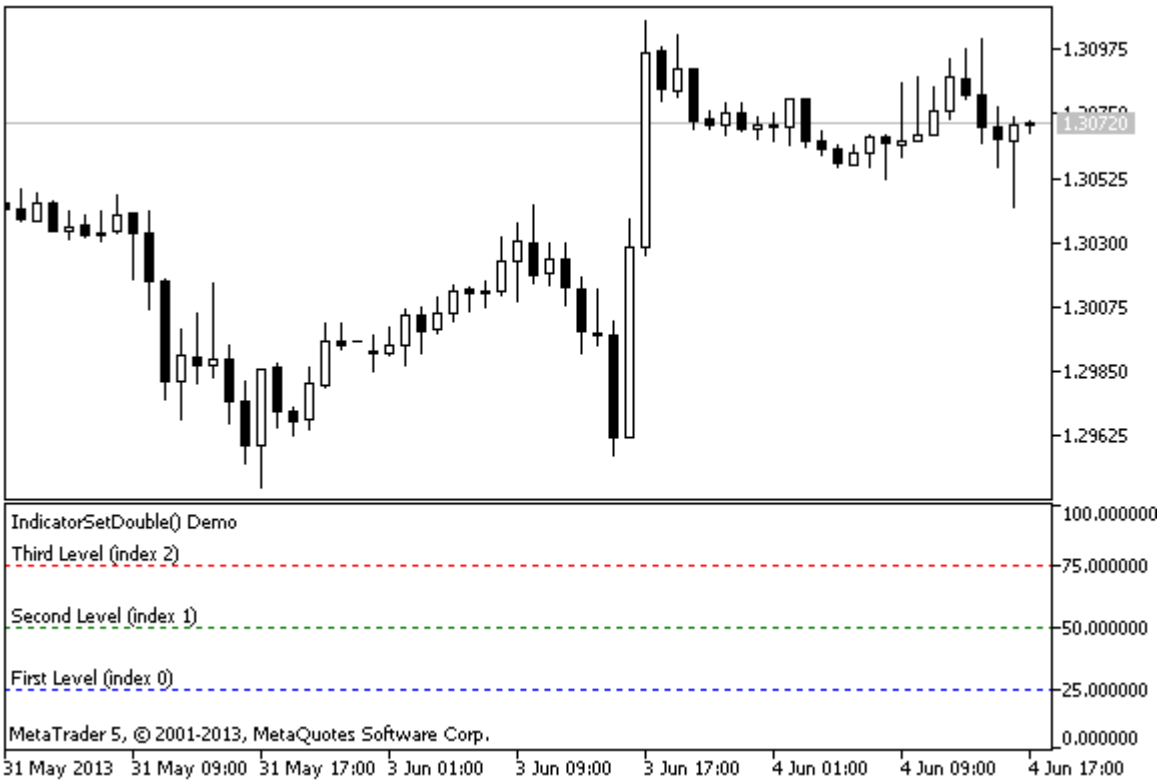
Not

#property direktifinin kullanımında özelliklerin numaralandırılması (şekillendiriciler) 1'den başlarken, fonksiyonlarda kullanılan numaralandırma 0'dan başlar. Seviye numarasının hatalı şekilde ayarlanmış olması durumunda, gösterge görünümü istenen görünümden farklı olabilir.

Örneğin, bir alt-penceredeki gösterge için ilk seviye değeri, iki farklı yolla ayarlanabilir:

- property indicator_level**1** 50 - 1 değeri seviye numarasını belirtmek için kullanılmıştır,
- IndicatorSetDouble(INDICATOR_LEVELVALUE, **0**, 50) - 0 değeri ilk seviyenin numarasını belirtmek için kullanılmıştır.

Örnek: gösterge penceresinin maksimum ve minimum değerlerini ve yatay çizgilerin konumlandırıldığı seviye değerlerini ters çeviren bir gösterge.




```
#property indicator_separate_window
//--- gösterge penceresinin maksimum ve minimum değerlerini ayarla
#property indicator_minimum  0
#property indicator_maximum  100
//--- ayrı bir gösterge penceresinde üç yatay seviye görüntüle
#property indicator_level1 25
#property indicator_level2 50
#property indicator_level3 75
//--- yatay seviyelerin kalınlığını ayarla
#property indicator_levelwidth 1
//--- yatay seviyelerin stilini ayarla
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- yatay seviyelerin açıklamalarını ayarla
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"İlk seviye (0 indisli)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"İkinci seviye (1 indisli)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Üçüncü seviye (2 indisli)");
//--- gösterge için kısa isim ayarla
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetDouble() Demo");
//--- her bir seviye için renk ayarla
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,0,clrBlue);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,1,clrGreen);
   IndicatorSetInteger(INDICATOR_LEVELCOLOR,2,clrRed);
```

```
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- tikleri hesapla
   tick_counter++;
//--- her 10-uncu tikle birlikte seviyeleri ters çevir
   if(tick_counter%10==0)
     {
      //--- seviye değerlerinin işaretlerini ters çevir
      level1=-level1;
      level2=-level2;
      level3=-level3;
      //--- maksimum ve minimum değerlerin işaretlerini ters çevir
      max-=shift;
      min-=shift;
      //--- kaydırma değerini ters çevir
      shift=-shift;
      //--- yeni seviye değerlerini ayarla
      IndicatorSetDouble(INDICATOR_LEVELVALUE,0,level1);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,1,level2);
      IndicatorSetDouble(INDICATOR_LEVELVALUE,2,level3);
      //--- gösterge penceresinin yeni maksimum ve minimum değerlerini ayarla
      Print("Başlat max = ",max,",   min = ",min);
      IndicatorSetDouble(INDICATOR_MAXIMUM,max);
      IndicatorSetDouble(INDICATOR_MINIMUM,min);
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Ayrıca Bakınız

Örneklerle Gösterge Stilleri, Gösterge Özellikleri ve Fonksiyonları arasındaki İlişki, Çizim Stilleri

IndicatorSetInteger

Karşılık gelen gösterge özelliğinin değerini ayarlar. Gösterge özelliği, int veya color tipinde olmalıdır. Fonksiyonun iki çeşidi bulunmaktadır.

Özellik tanımlayıcısı (kimliği) ile çağrı.

```
bool  IndicatorSetInteger(
   int  prop_id,           // tanımlayıcı
   int  prop_value         // ayarlanacak değer
   );
```

Özellik tanımlayıcısının ve şekillendiricinin belirtilmesi ile yapılan çağrı.

```
bool  IndicatorSetInteger(
   int  prop_id,           // tanımlayıcı
   int  prop_modifier,     // şekillendirici
   int  prop_value         // ayarlanacak değer
   )
```

Parametreler

prop_id

[in] Gösterge özelliğinin tanımlayıcısı. Bu değer, ENUM_CUSTOMIND_PROPERTY_INTEGER sayımının değerlerinden biri olabilir.

prop_modifier

[in] Belirtilen özelliğin şekillendiricisi. Sadece seviye özellikleri bir şekillendirici gerektirir.

prop_value

[in] Özellik değeri.

Dönüş değeri

Başarılı sonuç durumunda 'true', aksi durumda 'false' değerine dönüş yapar.

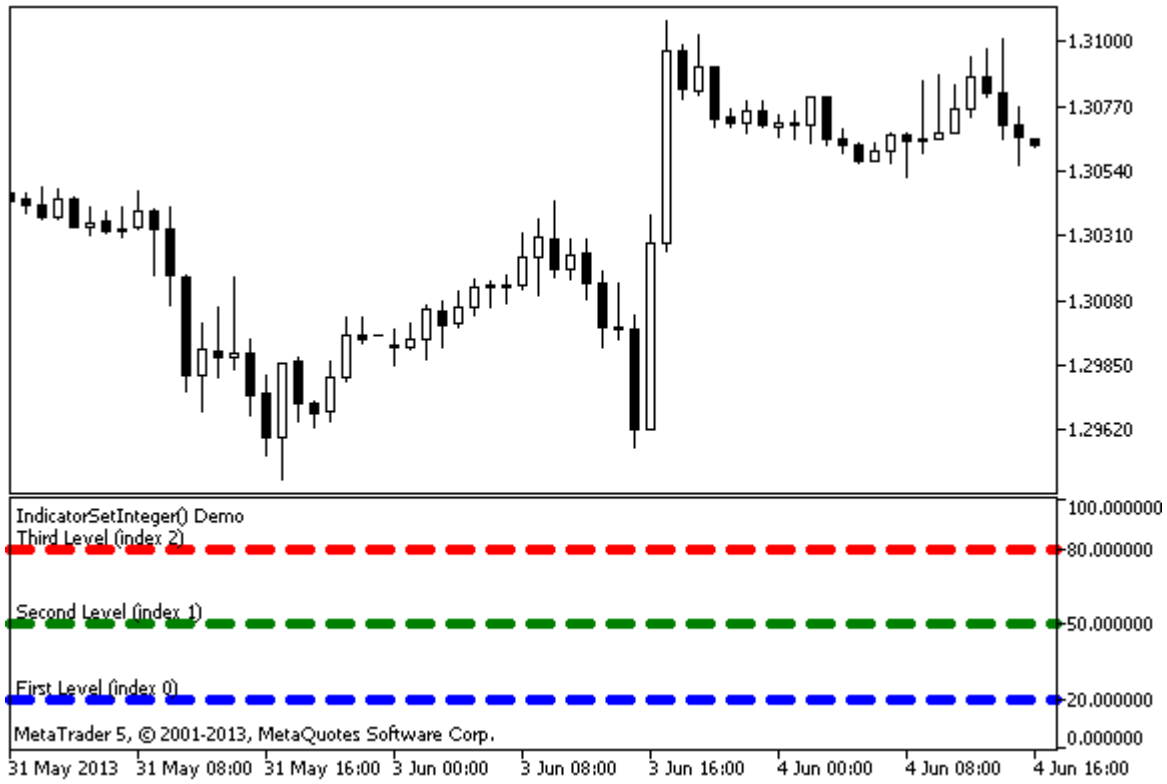
Not

#property direktifinin kullanımında özelliklerin numaralandırılması (şekillendiriciler) 1'den başlarken, fonksiyonlarda kullanılan numaralandırma 0'dan başlar. Seviye numarasının hatalı şekilde ayarlanmış olması durumunda, gösterge görünümü istenen görünümden farklı olabilir.

Örneğin, ilk yatay çizginin kalınlığını ayarlamak için, sıfırıncı indisi kullanın:

- IndicatorSetInteger(INDICATOR_LEVELWIDTH, **0**, 5) - 0 indisi, ilk seviyenin kalınlığını ayarlamak için kullanılır.

Örnek: yatay çizgilerin rengini, stilini ve kalınlığını ayarlayan gösterge.




```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- ayrı bir gösterge penceresinde üç yatay seviye görüntüle
#property indicator_level1 20
#property indicator_level2 50
#property indicator_level3 80
//--- yatay seviyelerin kalınlığını ayarla
#property indicator_levelwidth 5
//--- yatay seviyelerin rengini ayarla
#property indicator_levelcolor clrAliceBlue
//--- yatay seviyelerin stilini ayarla
#property indicator_levelstyle STYLE_DOT
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- yatay seviyelerin açıklamalarını ayarla
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"İlk seviye (0 indisli)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"İkinci seviye (1 indisli)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Üçüncü seviye (2 indisli)");
//--- gösterge için kısa isim ayarla
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetInteger() Demo");
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
```

```
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- tikleri hesapla
   tick_counter++;
//--- ve tik sayacına göre yatay seviyeleri ayarla
   ChangeLevelColor(0,tick_counter,3,6,10); // Son üç renk parametresi açılır
   ChangeLevelColor(1,tick_counter,3,6,8);
   ChangeLevelColor(2,tick_counter,4,7,9);
//--- yatay seviyelerin stilini değiştir
   ChangeLevelStyle(0,tick_counter);
   ChangeLevelStyle(1,tick_counter+5);
   ChangeLevelStyle(2,tick_counter+15);
//--- kalınlığı, tik sayısının 5'e bölümünden kalan şeklinde al
   int width=tick_counter%5;
//--- tüm yatay seviyelerde tekrarla ve kalınlığı ayarla
   for(int l=0;l<3;l++)
      IndicatorSetInteger(INDICATOR_LEVELWIDTH,l,width+1);
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| Ayrı gösterge penceresindeki yatay çizginin rengini ayarla       |
//+------------------------------------------------------------------+
void ChangeLevelColor(int level,      // yatay çizginin numarası
                      int tick_number,// bölüm, bölmenin kalanını alacak değer
                      int f_trigger,  // renk değişimi için ilk bölen
                      int s_trigger,  // renk değişimi için ikinci bölen
                      int t_trigger)  // renk değişimi için üçüncü bölen
  {
   static color colors[3]={clrRed,clrBlue,clrGreen};
//--- colors[] dizisinden renk indisi
   int index=-1;
//--- yatay çizginin boyanması için colors[] dizisindeki renk numarasını hesapla
   if(tick_number%f_trigger==0)
      index=0;   // eğer tick_number, f_trigger ile kalansız bölünüyorsa
   if(tick_number%s_trigger==0)
      index=1;   // eğer tick_number, s_trigger ile kalansız bölünüyorsa
```

```
   if(tick_number%t_trigger==0)
      index=2;    // eğer tick_number, t_trigger ile kalansız bölünüyorsa
//--- renk tanımlanmışsa ayarla
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELCOLOR,level,colors[index]);
//---
  }
//+------------------------------------------------------------------+
//| Ayrı gösterge penceresindeki yatay çizginin stilini ayarla       |
//+------------------------------------------------------------------+
void ChangeLevelStyle(int level,     // yatay çizginin numarası
                      int tick_number// bölümün kalanını almak için sayı
                      )
  {
//--- stillerin yükleneceği dizi
   static ENUM_LINE_STYLE styles[5]=
     {STYLE_SOLID,STYLE_DASH,STYLE_DOT,STYLE_DASHDOT,STYLE_DASHDOTDOT};
//--- styles[] dizisinden, stil indisi
   int index=-1;
//--- yatay çizginin stilini ayarlamak için, styles[] dizisinden sayıyı hesapla
   if(tick_number%50==0)
      index=5;    // tick_number, 50 ile kalansız bölünüyorsa, stil STYLE_DASHDOTDOT o
   if(tick_number%40==0)
      index=4;    // ... stil STYLE_DASHDOT olur
   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
//--- eğer stil tanımlanmışsa ayarla
   if(index!=-1)
      IndicatorSetInteger(INDICATOR_LEVELSTYLE,level,styles[index]);
  }
```

Ayrıca Bakınız

Özel Gösterge Özellikleri, Program Özellikleri (#property), Çizim Stilleri

IndicatorSetString

Karşılık gelen gösterge özelliğinin değerini ayarlar. Gösterge özelliği string tipinde olmalıdır. Fonksiyonun iki çeşidi bulunmaktadır.

Özellik tanımlayıcısı (kimliği) ile çağrı.

```
bool  IndicatorSetString(
   int     prop_id,           // tanımlayıcı
   string  prop_value         // ayarlanacak değer
   );
```

Özellik tanımlayıcısının ve şekillendiricinin belirtilmesi ile yapılan çağrı.

```
bool  IndicatorSetString(
   int     prop_id,           // tanımlayıcı
   int     prop_modifier,     // şekillendirici
   string  prop_value         // ayarlanacak değer
   )
```

Parametreler

prop_id

[in] Gösterge özelliğinin tanımlayıcısı. Bu değer, ENUM_CUSTOMIND_PROPERTY_STRING sayımının değerlerinden biri olabilir.

prop_modifier

[in] Belirtilen özelliğin şekillendiricisi. Sadece seviye özellikleri bir şekillendirici gerektirir.

prop_value

[in] Özellik değeri.

Dönüş değeri

Başarılı sonuç durumunda 'true', aksi durumda 'false' değerine dönüş yapar.

Not

#property direktifinin kullanımında özelliklerin numaralandırılması (şekillendiriciler) 1'den başlarken, fonksiyonlarda kullanılan numaralandırma 0'dan başlar. Seviye numarasının hatalı şekilde ayarlanmış olması durumunda, gösterge görünümü istenen görünümden farklı olabilir.

Örneğin, ilk yatay çizginin açıklamasını ayarlamak için, sıfırıncı indisi kullanın::

- IndicatorSetString(INDICATOR_LEVELTEXT, 0, "First Level") - 0 indisi, ilk seviyenin metin açıklamasını ayarlamak için kullanılır.

Örnek: gösterge yatay çizgilerine metin etiketleri ayarlayan gösterge.


```
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
//--- ayrı bir gösterge penceresinde üç yatay seviye görüntüle
#property indicator_level1 30
#property indicator_level2 50
#property indicator_level3 70
//--- yatay seviyelerin rengini ayarla
#property indicator_levelcolor clrRed
//--- yatay seviyelerin stilini ayarla
#property indicator_levelstyle STYLE_SOLID
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- yatay seviyelerin açıklamalarını ayarla
   IndicatorSetString(INDICATOR_LEVELTEXT,0,"İlk seviye (0 indisli)");
   IndicatorSetString(INDICATOR_LEVELTEXT,1,"İkinci seviye (1 indisli)");
   IndicatorSetString(INDICATOR_LEVELTEXT,2,"Üçüncü seviye (2 indisli)");
//--- gösterge için kısa isim ayarla
   IndicatorSetString(INDICATOR_SHORTNAME,"IndicatorSetString() Demo");
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
```

```
//+------------------------------------------------------------------+
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[])
  {
//---

//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

Ayrıca Bakınız

Özel Gösterge Özellikleri, Program Özellikleri (#property)

PlotIndexSetDouble

Belirtilen gösterge çizgisinin belirtilen özelliğinin değerini ayarlar. Gösterge özelliği double tipinde olmalıdır.

```
bool  PlotIndexSetDouble(
   int     plot_index,     // çizim stili indisi
   int     prop_id,        // özellik tanımlayıcı
   double  prop_value      // ayarlanacak değer
   );
```

Parametreler

plot_index

[in] grafiksel çizim indisi

prop_id

[in] Bu değer, ENUM_PLOT_PROPERTY_DOUBLE sayımının değerlerinden biri olabilir.

prop_value

[in] Özelliğin değeri.

Dönüş değeri

Başarılı sonuç durumunda true, aksi durumda false dönüşü yapar.

PlotIndexSetInteger

Belirtilen gösterge çizgisinin belirtilen özelliğinin değerini ayarlar. Gösterge özelliği int, char, bool veya color tiplerinden biri olmalıdır. Fonksiyonun 2 çeşidi bulunmaktadır.

Özellik tanımlayıcısını gösteren çağrı.

```
bool  PlotIndexSetInteger(
   int  plot_index,         // çizim stili index
   int  prop_id,            // özellik tanımlayıcısı
   int  prop_value          // ayarlanacak değer
   );
```

Özellik tanımlayıcısını ve şekillendiricisini gösteren çağrı.

```
bool  PlotIndexSetInteger(
   int  plot_index,         // çizim stili index
   int  prop_id,            // özellik tanımlayıcısı
   int  prop_modifier,      // özellik şekillendiricisi
   int  prop_value          // ayarlanacak değer
   )
```

Parametreler

plot_index

[in] grafiksel çizim indisi

prop_id

[in] Bu değer, ENUM_PLOT_PROPERTY_INTEGER sayımının değerlerinden biri olabilir.

prop_modifier

[in] Belirtilen özelliğin şekillendiricisi. Sadece renk indeksi özellikleri bir değiştirici gerektirir.

prop_value

[in] Özelliğin değeri.

Dönüş değeri

Başarılı sonuç durumunda true, aksi durumda false dönüşü yapar.

Örnek: üç renkli çizgi çizen bir gösterge örneği. Renk şeması her 5 tikte bir değişir.


```
#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
//--- gösterge tamponları
double         ColorLineBuffer[];
double         ColorBuffer[];
int            MA_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,ColorLineBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,ColorBuffer,INDICATOR_COLOR_INDEX);
//--- MA (Hareketli Ortalamanın) işleyicisini al
   MA_handle=iMA(Symbol(),0,10,0,MODE_EMA,PRICE_CLOSE);
//---
  }
//+------------------------------------------------------------------+
//| Renk indisini al                                                 |
//+------------------------------------------------------------------+
```

```
int getIndexOfColor(int i)
  {
   int j=i%300;
   if(j<100) return(0);// ilk indis
   if(j<200) return(1);// ikinci indis
   return(2); // üçüncü indis
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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;
//--- ilk hesaplama veya çubuk sayısı değişti
   if(prev_calculated==0)
     {
      //--- MA değerlerini ColorLineBuffer gösterge tamponuna kopyala
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);// kopyalama başarısız - çöpe at
      //--- şimdi her çubuk için çizgi rengini ayarla
      for(int i=0;i<rates_total;i++)
         ColorBuffer[i]=getIndexOfColor(i);
     }
   else
     {
      //--- MA değerlerini ColorLineBuffer gösterge tamponuna kopyala
      int copied=CopyBuffer(MA_handle,0,0,rates_total,ColorLineBuffer);
      if(copied<=0) return(0);

      ticks++;// tiklerin sayılması
      if(ticks>=5)// renk şemasının değişme vakti
        {
         ticks=0; // sayacı sıfırla
         modified++; // renk değişimi sayacı
         if(modified>=3)modified=0;// sayacı sıfırla
         ResetLastError();
         switch(modified)
           {
```

```
            case 0:// ilk renk şeması
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrRed);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrBlue);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrGreen);
               Print("Renk şeması "+modified);
               break;
            case 1:// ikinci renk şeması
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrYellow);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrPink);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLightSlateGray);
               Print("Renk şeması "+modified);
               break;
            default:// üçüncü renk şeması
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,0,clrLightGoldenrod);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,1,clrOrchid);
               PlotIndexSetInteger(0,PLOT_LINE_COLOR,2,clrLimeGreen);
               Print("Renk şeması "+modified);
           }
        }
      else
        {
         //--- başlangıç konumunu ayarla
         limit=prev_calculated-1;
         //--- şimdi her çubuk için çizgi rengini ayarlıyoruz
         for(int i=limit;i<rates_total;i++)
            ColorBuffer[i]=getIndexOfColor(i);
        }
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

PlotIndexSetString

Belirtilen gösterge çizgisinin belirtilen özelliğinin değerini ayarlar. Gösterge özelliği string tipinde olmalıdır.

```
bool  PlotIndexSetString(
   int     plot_index,     // çizim stili indisi
   int     prop_id,        // özellik tanımlayıcı
   string  prop_value      // ayarlanacak değer
   );
```

Parametreler

plot_index

[in] grafiksel çizim indisi

prop_id

[in] Bu değer ENUM_PLOT_PROPERTY_STRING sayımının değerlerinden biri olabilir.

prop_value

[in] Özelliğin değeri.

Dönüş değeri

Başarılı sonuç durumunda true, aksi durumda false dönüşü yapar.

PlotIndexGetInteger

Belirtilen gösterge çizgisinin belirtilen özelliğinin değerini ayarlar. Gösterge özelliği int, color, bool veya char tiplerinden biri olmalıdır. Fonksiyonun 2 çeşidi bulunmaktadır.

Özellik tanımlayıcısını gösteren çağrı.

```
int  PlotIndexGetInteger(
   int  plot_index,         // çizim stili index
   int  prop_id,            // özellik tanımlayıcısı
   );
```

Özellik tanımlayıcısını ve şekillendiricisini gösteren çağrı.

```
int  PlotIndexGetInteger(
   int  plot_index,         // çizim indisi
   int  prop_id,            // özellik tanımlayıcısı
   int  prop_modifier       // özellik şekillendirici
   )
```

Parametreler

plot_index

[in] grafiksel çizim indisi

prop_id

[in] Bu değer, ENUM_PLOT_PROPERTY_INTEGER sayımının değerlerinden biri olabilir.

prop_modifier

[in] Belirtilen özelliğin şekillendiricisi. Sadece renk indeksi özellikleri bir değiştirici gerektirir.

Not

Fonksiyon, uygun bir göstergenin çizim ayarlarını almak amacıyla tasarlanmıştır. Fonksiyon, bir çizginin çizim özelliklerini diğerine kopyalamak için PlotIndexSetInteger fonksiyonu ile birlikte çalışır.

Örnek: Haftanın günlerine göre mumları renklendiren bir gösterge. Her gün için renkler, bir program şekilde ayarlanır.


```
#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
//--- gösterge tamponları
double          OpenBuffer[];
double          HighBuffer[];
double          LowBuffer[];
double          CloseBuffer[];
double          ColorCandlesColors[];
color           ColorOfDay[6]={CLR_NONE,clrMediumSlateBlue,
                               clrDarkGoldenrod,clrForestGreen,clrBlueViolet,clrRed};
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   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);
//--- color tamponundaki renklerin numarasını ayarla
```

```
   PlotIndexSetInteger(0,PLOT_COLOR_INDEXES,6);
//--- color tamponu için renkleri
   for(int i=1;i<6;i++)
      PlotIndexSetInteger(0,PLOT_LINE_COLOR,i,ColorOfDay[i]);
//--- kesinliği ayarla
   IndicatorSetInteger(INDICATOR_DIGITS,_Digits);
   printf("%u gün rengine sahibiz",PlotIndexGetInteger(0,PLOT_COLOR_INDEXES));
//---
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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];
      //--- her mum için renk ayarla
      TimeToStruct(time[i],t);
      ColorCandlesColors[i]=t.day_of_week;
      //---
      i++;
     }
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

Nesne Fonksiyonları

Bu, belirtilen herhangi bir çizelge için grafiksel nesnelerle çalışmak amacıyla geliştirilmiş fonksiyonlardan oluşan bir gruptur.

Nesnelerin oluşturulması ve çizelge üzerinde taşınması için tasarlanmış ObjectCreate() ve ObjectMove() işlevleri gibi, grafiksel nesnelerin özelliklerini tanımlayan fonksiyonlar da, çizelgeye komutlar göndermek için kullanılır. Bu fonksiyonlar başarılı şekilde çalıştırılırsa, söz konusu değişim komutu çizelge olaylarının genel kuyruğuna eklenir. Çizelge olayları kuyruğu işlendiğinde, grafiksel nesne özelliklerindeki görsel değişimler çizelgeye uygulanır.

Bu sebepten ötürü, bu fonksiyonların çağrılmalarının hemen ardından grafiksel nesneler üzerinde bir güncelleme beklememeniz gerekir. Grafiksel nesneler, genellikle değişim olaylarını takiben, terminal tarafından otomatik olarak güncellenir - yeni bir fiyat teklifinin gelmesinden sonra, çizelge penceresinin yeniden boyutlandırılmasının ardından, vb. Grafiksel nesneleri zorla güncellemek için ChartRedraw() fonksiyonunu kullanın.

Fonksiyon	Eylem
ObjectCreate	Belirtilen çizelge üzerinde belirlenen tipte bir nesne oluşturur
ObjectName	Belirtilen çizelge (veya çizelge alt-penceresi) üzerinde belirtilen tipte bir nesnenin ismine dönüş yapar
ObjectDelete	Belirtilen çizelge (veya çizelge alt-penceresi) üzerinde belirtilen tipte bir nesneyi siler
ObjectsDeleteAll	Belirtilen çizelge (veya çizelge alt-penceresi) üzerinde belirtilen tipteki tüm nesneleri siler
ObjectFind	Belirtilen tanımlayıcı ve ismi kullanarak bir nesneyi arar
ObjectGetTimeByValue	Nesnenin belirtilen fiyat değeri için zaman değerine dönüş yapar
ObjectGetValueByTime	Nesnenin belirtilen zaman değeri için fiyat değerine dönüş yapar
ObjectMove	Bir nesnenin tutturma noktası koordinatlarını değiştirir
ObjectsTotal	Belirtilen çizelge (veya çizelge alt-penceresi) üzerinde belirtilen tipteki nesnelerin sayısına dönüş yapar
ObjectGetDouble	Karşılık gelen nesne özelliğinin double tipi değerine dönüş yapar
ObjectGetInteger	Karşılık gelen nesne özelliğinin tamsayı değerine dönüş yapar
ObjectGetString	Karşılık gelen nesne özelliğinin string tipli değerine dönüş yapar
ObjectSetDouble	Karşılık gelen nesne özelliğinin değerini ayarlar
ObjectSetInteger	Karşılık gelen nesne özelliğinin değerini ayarlar
ObjectSetString	Karşılık gelen nesne özelliğinin değerini ayarlar
TextSetFont	Çizim yöntemlerini kullanarak, görüntülenecek yazı tipini ayarlar (varsayılan olarak Arial 20 kullanılır)

Fonksiyon	Eylem
TextOut	Bir metni, grafiksel kaynak oluşturmak amacıyla tasarlanmış özel bir diziye (tampona) aktarır
TextGetSize	Mevcut yazı tipi ayarları için dizginin genişlik ve yükseklik değerlerine dönüş yapar

Her grafiksel nesne, tutturulduğu çizelge içinde (alt-pencereler de dahil olmak üzere) benzersiz bir isme sahip olmalıdır. Bir grafiksel nesnenin isminin değiştirilmesi sonucunda iki olay ortaya çıkar: eski isimli nesnenin silinmesi olayı ve yeni isimli nesnenin oluşturulması olayı.

Bir nesnenin oluşturulmasının veya bir nesne özelliğinin değiştirilmesinin ardından ChartRedraw() fonksiyonunun kullanılması önerilir. Bu fonksiyon, terminale, çizelgenin (ve üstündeki diğer tüm görünür nesnelerin) zorla yeniden çizilmesi komutunu verir.

ObjectCreate

Belirtilen çizelge alt-penceresi üzerinde, belirtilen isim, tip ve koordinatlar ile yeni bir nesne oluşturur. Nesnenin oluşturulması sırasında 30 koordinat belirtilebilir.

```
bool  ObjectCreate(
   long          chart_id,      // çizelge tanımlayıcısı
   string        name,          // nesne ismi
   ENUM_OBJECT   type,          // nesne tipi
   sub_window    nwin,          // pencere indisi
   datetime      time1,         // ilk tutturma noktasının zaman koordinatı
   double        price1,        // ilk tutturma noktasının fiyat koordinatı
  ...
   datetime      timeN=0,       // N-inci tutturma noktasının zaman koordinatı
   double        priceN=0,      // N-inci tutturma noktasının fiyat koordinatı
  ...
   datetime      time30=0,      // 30-uncu tutturma noktasının zaman koordinatı
   double        price30=0      // 30-uncu tutturma noktasının fiyat koordinatı
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

name

[in] Nesne ismi. Bu isim, çizelge (alt-pencereler de dahil) içinde benzersiz olmalıdır.

type

[in] Nesne tipi. Bu değer, ENUM_OBJECT sayımının değerlerinden biri olabilir.

sub_window

[in] Çizelge alt penceresinin indisi. 0, ana çizelge penceresi demektir. Belirtilen alt-pencere mevcut olmalıdır aksi durumda fonksiyon 'false' dönüşü yapar.

time1

[in] İlk tutturma noktasının zaman koordinatı.

price1

[in] İlk tutturma noktasının fiyat koordinatı.

timeN=0

[in] N-inci tutturma noktasının zaman koordinatı.

priceN=0

[in] N-inci tutturma noktasının fiyat koordinatı.

time30=0

[in] Otuzuncu tutturma noktasının zaman koordinatı.

price30=0

[in] Otuzuncu tutturma noktasının fiyat koordinatı.

Dönüş değeri

Fonksiyon, komutun doğru olarak belirtilen grafiğin sırasına başarıyla eklendiğinde true, aksi halde false döndürür. Bir nesne zaten oluşturulmuşsa, onun koordinatlarını değiştirmek için bir girişim yapılır.

Not

ObjectCreate() için her zaman eşzamansız bir çağrı kullanılır, bu nedenle fonksiyon yalnızca komutu bir grafik sırasına ekleme sonuçlarını verir. Bu durumda true yalnızca komutun başarıyla eklendiği anlamına gelir, ancak yürütme sonucu bilinmemektedir.

Komut yürütme sonucunu kontrol etmek için ObjectFind() işlevini veya ObjectGetXXX gibi nesne özelliklerini isteyen başka herhangi bir işlevi kullanabilirsiniz. Bununla birlikte, bu tür işlevlerin bu grafiğin kuyruğunun sonuna eklendiğini ve yürütme sonucunu beklediğini (senkron çağrı nedeniyle) unutmamalısınız ve bu nedenle zaman alıcı olabilir. Bu özellik, bir grafikte çok sayıda nesneyle çalışırken dikkate alınmalıdır.

Bir nesne adı 63 karakteri aşmamalıdır.

Çizelge alt-pencerelerinin numaralandırılmasına (eğer göstergeli alt-pencereler mevcutsa) 1 ile başlanır. Ana çizelge penceresi her zaman 0 indisine sahiptir.

Fazla sayıda (30'a kadar) tutturma noktası, ilerideki kullanımlar için uygulanır. Ayrıca, grafiksel nesneler için kullanılacak 30 muhtemel tutturma noktası, bir fonksiyonun çağrısında kullanılabilecek parametrelerin sayısının sınırına (64'ten fazla olamaz) bağlı olarak belirlenir.

Bir nesne yeniden isimlendirildiğinde, aynı anda iki olay ortaya çıkar. Bu olaylar, bir Uzman Danışman veya gösterge içerisinde OnChartEvent() fonksiyonu ile işlenebilir:

- eski isimli nesnenin silinmesi olayı;
- yeni isimli nesnenin oluşturulması olayı.

Her bir nesne tipi için belirtilmesi gereken belli sayıda tutturma noktası vardır:

Tanıtıcı	Açıklama	Tutturma noktaları
OBJ_VLINE	Dikey Çizgi	Tek tutturma noktası. Gerçekte, sadece zaman koordinatı kullanılır.
OBJ_HLINE	Yatay Çizgi	Tek tutturma noktası. Gerçekte, sadece fiyat koordinatı kullanılır.<
OBJ_TREND	Trend Çizgisi	İki tutturma noktası.
OBJ_TRENDBYANGLE	Açılı Trend Çizgisi	İki tutturma noktası.
OBJ_CYCLES	Döngü Çizgileri	İki tutturma noktası.
OBJ_ARROWED_LINE	Oklu çizgi	İki tutturma noktası.
OBJ_CHANNEL	Eşit-Aralıklı Kanal	Üç tutturma noktası.
OBJ_STDDEVCHANNEL	Standart Sapma Kanalı	İki tutturma noktası.
OBJ_REGRESSION	Doğrusal Regresyon Kanalı	İki tutturma noktası.

Tanıtıcı	Açıklama	Tutturma noktaları
OBJ_PITCHFORK	Andrews Çatalı (Üçlü Çizgi)	Üç tutturma noktası.
OBJ_GANNLINE	Gann Çizgisi	İki tutturma noktası.
OBJ_GANNFAN	Gann Yelpazesi	İki tutturma noktası.
OBJ_GANNGRID	Gann Izgarası	İki tutturma noktası.
OBJ_FIBO	Fibonacci Düzeltme Seviyeleri	İki tutturma noktası.
OBJ_FIBOTIMES	Fibonacci Zaman Dilimleri	İki tutturma noktası.
OBJ_FIBOFAN	Fibonacci Yelpazesi	İki tutturma noktası.
OBJ_FIBOARC	Fibonacci Yayları	İki tutturma noktası.
OBJ_FIBOCHANNEL	Fibonacci Kanalı	Üç tutturma noktası.
OBJ_EXPANSION	Fibonacci Açılımı	Üç tutturma noktası.
OBJ_ELLIOTWAVE5	Elliott Dürtü Dalgası	Beş tutturma noktası.
OBJ_ELLIOTWAVE3	Elliott Düzeltme Dalgası	Üç tutturma noktası.
OBJ_RECTANGLE	Dikdörtgen	İki tutturma noktası.
OBJ_TRIANGLE	Üçgen	Üç tutturma noktası.
OBJ_ELLIPSE	Elips	Üç tutturma noktası.
OBJ_ARROW_THUMB_UP	İyi (Başparmak Yukarı)	Tek tutturma noktası.
OBJ_ARROW_THUMB_DOWN	Kötü (Başparmak Aşağı)	Tek tutturma noktası.
OBJ_ARROW_UP	Yukarı Ok	Tek tutturma noktası.
OBJ_ARROW_DOWN	Aşağı Ok	Tek tutturma noktası.
OBJ_ARROW_STOP	Dur (Stop) İşareti	Tek tutturma noktası.
OBJ_ARROW_CHECK	Kontrol İşareti	Tek tutturma noktası.
OBJ_ARROW_LEFT_PRICE	Sol Fiyat Etiketi	Tek tutturma noktası.
OBJ_ARROW_RIGHT_PRICE	Sağ Fiyat Etiketi	Tek tutturma noktası.
OBJ_ARROW_BUY	Alış Sinyali	Tek tutturma noktası.
OBJ_ARROW_SELL	Satış Sinyali	Tek tutturma noktası.
OBJ_ARROW	Ok	Tek tutturma noktası.
OBJ_TEXT	Metin	Tek tutturma noktası.
OBJ_LABEL	Etiket	Nesne konumu, OBJPROP_XDISTANCE ve OBJPROP_YDISTANCE özellikleri ile belirlenir.

Tanıtıcı	Açıklama	Tutturma noktaları
OBJ_BUTTON	Düğme	Nesne konumu, OBJPROP_XDISTANCE ve OBJPROP_YDISTANCE özellikleri ile belirlenir.
OBJ_CHART	Çizelge	Nesne konumu, OBJPROP_XDISTANCE ve OBJPROP_YDISTANCE özellikleri ile belirlenir.
OBJ_BITMAP	Bitmap Etiketi	Tek tutturma noktası.
OBJ_BITMAP_LABEL	Bitmap Etiketi	Nesne konumu, OBJPROP_XDISTANCE ve OBJPROP_YDISTANCE özellikleri ile belirlenir.
OBJ_EDIT	Düzenle	Nesne konumu, OBJPROP_XDISTANCE ve OBJPROP_YDISTANCE özellikleri ile belirlenir.
OBJ_EVENT	Ekonomik takvimdeki bir olaya karşılık gelen "Olay" nesnesi	Tek tutturma noktası. Gerçekte, sadece zaman koordinatı kullanılır.
OBJ_RECTANGLE_LABEL	Özel grafiksel arayüzler oluşturmak ve tasarlamak için "Dikdörtgen Etiket" nesnesi.	Nesne konumu, OBJPROP_XDISTANCE ve OBJPROP_YDISTANCE özellikleri ile belirlenir.

ObjectName

Belirtilen çizelgenin belirtilen alt-penceresinde, karşılık gelen belirli tipteki nesnenin ismine dönüş yapar.

```
string  ObjectName(
   long  chart_id,           // çizelge tanımlayıcısı
   int   pos,                // nesne listesindeki numara
   int   sub_window=-1,      // pencere indisi
   int   type=-1             // nesne tipi
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

pos

[in] Numara, tip ve alt-pencere indisi ile belirtilen filtreye göre, nesnenin sıra numarası.

sub_window=-1

[in] Çizelge alt penceresinin indisi. 0 ana çizelge penceresidir, -1 ise ana pencere de dahil olmak üzere tüm alt pencereler anlamına gelir.

type=-1

[in] Nesne tipi. Bu değer, ENUM_OBJECT değerlerinden biri olabilir. -1 tüm tipler anlamına gelir.

Dönüş değeri

Başarı durumunda nesnenin ismine dönüş yapılır.

Not

Fonksiyon, eşzamanlı bir çağrı kullanır; bu, fonksiyonun çağrı öncesinde bu grafik için yerine getirilmiş tüm komutların yürütülmesini beklediği anlamına gelir, bu nedenle bu fonksiyon çok zaman alıcı olabilir. Bu özellik, bir grafikte çok sayıda nesneyle çalışırken dikkate alınmalıdır.

Bir nesne yeniden isimlendirildiğinde, aynı anda iki olay ortaya çıkar. Bu olaylar, bir Uzman Danışman veya gösterge içerisinde OnChartEvent() fonksiyonu ile işlenebilir:

- eski isimli nesnenin silinmesi olayı;
- yeni isimli nesnenin oluşturulması olayı.

ObjectDelete

Belirtilen sembolü belirtilen çizelgeden kaldırır.

```
bool  ObjectDelete(
   long    chart_id,      // çizelge tanımlayıcı
   string  name           // nesne ismi
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

name

[in] Silinecek nesnenin ismi.

Dönüş değeri

Fonksiyon, komutun doğru olarak belirtilen grafiğin sırasına başarıyla eklendiğinde true, aksi halde false döndürür.

Not

Eşzamanlı olmayan bir çağrı her zaman ObjectDelete() için kullanılır, bu nedenle fonksiyon yalnızca komutu bir grafik sırasına ekleme sonuçlarını döndürür. Bu durumda true yalnızca komutun başarıyla eklendiği anlamına gelir, ancak yürütme sonucu bilinmemektedir.

Komut yürütme sonucunu kontrol etmek için ObjectFind() işlevini veya ObjectGetXXX gibi nesne özelliklerini isteyen başka herhangi bir işlevi kullanabilirsiniz. Bununla birlikte, bu tür işlevlerin bu grafiğin kuyruğunun sonuna eklendiğini ve yürütme sonucunu beklediğini (senkron çağrı nedeniyle) unutmamalısınız ve bu nedenle zaman alıcı olabilir. Bu özellik, bir grafikte çok sayıda nesneyle çalışırken dikkate alınmalıdır.

Bir nesne yeniden isimlendirildiğinde, aynı anda iki olay ortaya çıkar. Bu olaylar, bir Uzman Danışman veya gösterge içerisinde OnChartEvent() fonksiyonu ile işlenebilir:

- eski isimli nesnenin silinmesi olayı;
- yeni isimli nesnenin oluşturulması olayı.

ObjectsDeleteAll

Belirtilen çizelge (veya çizelge alt-penceresi) üzerinde belirtilen tipteki tüm nesneleri siler

```
int  ObjectsDeleteAll(
   long  chart_id,           // çizelge tanımlayıcısı
   int   sub_window=-1,      // pencere indisi
   int   type=-1             // nesne tipi
   );
```

Belirtilen türdeki tüm nesneleri nesne adlarındaki önek kullanarak kaldırır.

```
int  ObjectsDeleteAll(
   long           chart_id,   // chart ID
   const string     prefix,   // nesne adındaki önek
   int        sub_window=-1,  // pencere indeksi
   int      object_type=-1    // nesne türü
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

prefix

[in] Nesne isimlerindeki önek. Adı bu karakter kümesiyle başlayan tüm nesneler grafikten kaldırılır. Öneki 'ad' veya 'ad*' olarak belirtebilirsiniz - her iki türe de aynı işe yarayacaktır. Önek olarak boş bir string belirtilirse, olası tüm adlara sahip nesneler kaldırılır.

sub_window=-1

[in] Çizelge alt-penceresinin numarası. 0 ana çizelge penceresidir, -1 ise ana pencere de dahil olmak üzere tüm alt pencereler anlamına gelir.

type=-1

[in] Nesne tipi. Bu değer, ENUM_OBJECT değerlerinden biri olabilir. -1 tüm tipler anlamına gelir.

Dönüş değeri

Silinen nesnelerin sayısına dönüş yapar. Hata ile ilgili daha fazla bilgi için GetLastError() fonksiyonunu çağırın.

Not

Fonksiyon, eşzamanlı bir çağrı kullanır; bu, fonksiyonun çağrı öncesinde bu grafik için yerine getirilmiş tüm komutların yürütülmesini beklediği anlamına gelir, bu nedenle bu fonksiyon çok zaman alıcı olabilir. Bu özellik, bir grafikte çok sayıda nesneyle çalışırken dikkate alınmalıdır.

ObjectFind

Belirtilen tanımlayıcıyı ve ismi kullanarak bir nesneyi çizelgede arar

```
int  ObjectFind(
   long    chart_id,      // çizelge tanımlayıcı
   string  name           // nesne ismi
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

name

[in] Aranan nesnenin ismi.

Dönüş değeri

Başarılı olunması durumunda, nesnenin bulunduğu alt-pencerenin numarasına dönüş yapar (0 ana pencere demektir). Nesne bulunamazsa fonksiyon, bir negatif sayıya dönüş yapar. Hata ile ilgili daha fazla bilgi için GetLastError() fonksiyonunu çağırın.

Not

Fonksiyon, eşzamanlı bir çağrı kullanır; bu, fonksiyonun çağrı öncesinde bu grafik için yerine getirilmiş tüm komutların yürütülmesini beklediği anlamına gelir, bu nedenle bu fonksiyon çok zaman alıcı olabilir. Bu özellik, bir grafikte çok sayıda nesneyle çalışırken dikkate alınmalıdır.

Bir nesne yeniden isimlendirildiğinde, aynı anda iki olay ortaya çıkar. Bu olaylar, bir Uzman Danışman veya gösterge içerisinde OnChartEvent() fonksiyonu ile işlenebilir:

- eski isimli nesnenin silinmesi olayı;
- yeni isimli nesnenin oluşturulması olayı.

ObjectGetTimeByValue

Nesnenin belirtilen fiyat değeri için zaman değerine dönüş yapar.

```
datetime  ObjectGetTimeByValue(
   long     chart_id,      // çizelge tanımlayıcı
   string   name,          // nesne ismi
   double   value,         // Fiyat
   int      line_id        // Çizgi numarası
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

name

[in] Nesne ismi.

value

[in] Fiyat değeri.

line_id

[in] Çizgi tanımlayıcı.

Dönüş değeri

Belirtilen nesnenin belirtilen fiyat değeri için zaman değerine dönüş yapar.

Not

Fonksiyon, eşzamanlı bir çağrı kullanır; bu, fonksiyonun çağrı öncesinde bu grafik için yerine getirilmiş tüm komutların yürütülmesini beklediği anlamına gelir, bu nedenle bu fonksiyon çok zaman alıcı olabilir. Bu özellik, bir grafikte çok sayıda nesneyle çalışırken dikkate alınmalıdır.

Bir nesne, bir fiyat koordinatında birden fazla değere sahip olabilir, bu yüzden çizgi numarasının da belirtilmesi gerekir. Bu fonksiyon sadece şu nesnelere uygulanır:

- Trend çizgisi (OBJ_TREND)
- Açılı trend çizgisi (OBJ_TRENDBYANGLE)
- Gann çizgisi (OBJ_GANNLINE)
- Eşit aralıklı kanal (OBJ_CHANNEL) - 2 çizgi
- Doğrusal regresyon kanalı (OBJ_REGRESSION) - 3 çizgi
- Standart sapma kanalı (OBJ_STDDEVCHANNEL) - 3 çizgi
- Oklu çizgi (OBJ_ARROWED_LINE)

Ayrıca Bakınız

Nesne Tipleri

ObjectGetValueByTime

Nesnenin belirtilen zaman değeri için fiyat değerine dönüş yapar.

```
double  ObjectGetValueByTime(
   long      chart_id,     // çizelge tanımlayıcı
   string    name,         // nesne ismi
   datetime  time,         // Zaman
   int       line_id       // Çizgi numarası
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

name

[in] Nesne ismi.

time

[in] Zaman değeri.

line_id

[in] Çizgi tanımlayıcı.

Dönüş değeri

Belirtilen nesnenin belirtilen zaman değeri için fiyat değerine dönüş yapar.

Not

Fonksiyon, eşzamanlı bir çağrı kullanır; bu, fonksiyonun çağrı öncesinde bu grafik için yerine getirilmiş tüm komutların yürütülmesini beklediği anlamına gelir, bu nedenle bu fonksiyon çok zaman alıcı olabilir. Bu özellik, bir grafikte çok sayıda nesneyle çalışırken dikkate alınmalıdır.

Bir nesne, bir fiyat koordinatında birden fazla değere sahip olabilir, bu yüzden çizgi numarasının da belirtilmesi gerekir. Bu fonksiyon sadece şu nesnelere uygulanır:

- Trend çizgisi (OBJ_TREND)
- Açılı trend çizgisi (OBJ_TRENDBYANGLE)
- Gann çizgisi (OBJ_GANNLINE)
- Eşit aralıklı kanal (OBJ_CHANNEL) - 2 çizgi
- Doğrusal regresyon kanalı (OBJ_REGRESSION) - 3 çizgi
- Standart sapma kanalı (OBJ_STDDEVCHANNEL) - 3 çizgi
- Oklu çizgi (OBJ_ARROWED_LINE)

Ayrıca Bakınız

Nesne Tipleri

ObjectMove

Nesnenin belirtilen tutturma noktası koordinatını değiştirir.

```
bool  ObjectMove(
   long      chart_id,         // çizelge tanımlayıcı
   string    name,             // nesne ismi
   int       point_index,      // tutturma noktası indisi
   datetime  time,             // Zaman
   double    price             // Fiyat
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

name

[in] Nesne ismi.

point_index

[in] Tutturma noktasının indisi. Tutturma noktalarının sayısı, nesnenin tipine bağlıdır.

time

[in] Seçilen tutturma noktasının zaman koordinatı.

price

[in] Seçilen tutturma noktasının fiyat koordinatı.<

Dönüş değeri

Fonksiyon, komutun doğru olarak belirtilen grafiğin sırasına başarıyla eklendiğinde true, aksi halde false döndürür.

Not

ObjectMove() için her zaman eşzamansız bir çağrı kullanılır, bu nedenle fonksiyon yalnızca komutun bir grafik sırasına eklenmesinin sonuçlarını döndürür. Bu durumda true yalnızca komutun başarıyla eklendiği anlamına gelir, ancak yürütme sonucu bilinmemektedir.

Komut yürütme sonucunu kontrol etmek için, ObjectGetXXX gibi nesne özelliklerini isteyen bir fonksiyon kullanabilirsiniz. Bununla birlikte, bu tür işlevlerin bu grafiğin kuyruğunun sonuna eklendiğini ve yürütme sonucunu beklediğini (senkron çağrı nedeniyle) unutmamalısınız ve bu nedenle zaman alıcı olabilir. Bu özellik, bir grafikte çok sayıda nesneyle çalışırken dikkate alınmalıdır.

ObjectsTotal

Belirtilen çizelgenin belirtilen alt-penceresinde, belirli tipteki nesnelerin sayısına dönüş yapar.

```
int  ObjectsTotal(
   long  chart_id,           // çizelge tanımlayıcısı
   int   sub_window=-1,      // pencere indisi
   int   type=-1             // nesne tipi
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

sub_window=-1

[in] Çizelge alt penceresinin indisi. 0 ana çizelge penceresidir, -1 ise ana pencere de dahil olmak üzere tüm alt pencereler anlamına gelir.

type=-1

[in] Nesne tipi. Bu değer, ENUM_OBJECT değerlerinden biri olabilir. -1 tüm tipler anlamına gelir.

Dönüş değeri

Nesnelerin sayısı.

Not

Fonksiyon, eşzamanlı bir çağrı kullanır; bu, fonksiyonun çağrı öncesinde bu grafik için yerine getirilmiş tüm komutların yürütülmesini beklediği anlamına gelir, bu nedenle bu fonksiyon çok zaman alıcı olabilir. Bu özellik, bir grafikte çok sayıda nesneyle çalışırken dikkate alınmalıdır.

ObjectSetDouble

Belirtilen nesne özelliğinin değerini ayarlar. Nesne özelliği double tipinde olmalıdır. Fonksiyonun iki çeşidi vardır.

Birinci versiyonda şekillendirici yer almaz

```
bool  ObjectSetDouble(
   long                          chart_id,          // çizelge tanımlayıcı
   string                        name,              // nesne ismi
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // özellik tanımlayıcısı
   double                        prop_value         // değer
   );
```

İkinci versiyonda değer, şekillendirici belirtilerek ayarlanır

```
bool  ObjectSetDouble(
   long                          chart_id,          // çizelge tanımlayıcı
   string                        name,              // nesne ismi
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // özellik tanımlayıcısı
   int                           prop_modifier,     // şekillendirici
   double                        prop_value         // değer
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

name

[in] Nesne ismi.

prop_id

[in] Nesne özelliğinin tanıtıcısı. Bu değer ENUM_OBJECT_PROPERTY_DOUBLE sayımının değerlerinden biri olabilir.

prop_modifier

[in] Belirtilen özelliğin şekillendiricisi. Fibonacci araçlarındaki ve Andrews Dirgenindeki seviye numarasını gösterir. Seviyelerin numaralandırılmasına sıfırdan başlanır.

prop_value

[in] Özelliğin değeri.

Dönüş değeri

Grafiksel nesnenin özelliğini değiştirmek için oluşturulan komut, çizelgeye başarılı şekilde gönderilmişse 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata ile ilgili daha fazla bilgi için GetLastError() fonksiyonunu çağırın.

Not

Fonksiyon, eşzamansız bir çağrı kullanır; bu, işlevin, belirtilen grafiğin sırasına eklenen komutun yürütülmesini beklemediği anlamına gelir. Bunun yerine, hemen kontrol döndürür.

Komut yürütme sonucunu kontrol etmek için, belirtilen nesne mülkiyetini isteyen bir fonksiyon kullanabilirsiniz. Bununla birlikte, bu tür işlevlerin bu çizelgenin sonuna eklendiğini ve yürütme sonucunu beklediklerini ve bu nedenle zaman alıcı olabileceğini unutmamalısınız. Bu özellik, bir grafikte çok sayıda nesneyle çalışırken dikkate alınmalıdır.

Bir fibonacci nesnesinin oluşturulmasını ve ona bir seviye eklenmesini gösteren örnek

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- yardımcı diziler
   double high[],low[],price1,price2;
   datetime time[],time1,time2;
//--- Açılış fiyatlarını kopyala - sondan 100 çubuk yeterli
   int copied=CopyHigh(Symbol(),0,0,100,high);
   if(copied<=0)
     {
      Print("High (yüksek) fiyat serisi kopyalanamadı");
      return;
     }
//--- Kapanış fiyatlarını kopyala - sondan 100 çubuk yeterli
   copied=CopyLow(Symbol(),0,0,100,low);
   if(copied<=0)
     {
      Print("Low (düşük) fiyat serisi kopyalanamadı");
      return;
     }
//--- Son 100 çubuk için açılış zamanını kopyala
   copied=CopyTime(Symbol(),0,0,100,time);
   if(copied<=0)
     {
      Print("Time serisinin değerleri kopyalanamadı");
      return;
     }
//--- kopyalanan veriye erişim şeklini zaman-serilerindeki gibi ayarla - geriye doğru
   ArraySetAsSeries(high,true);
   ArraySetAsSeries(low,true);
   ArraySetAsSeries(time,true);

//--- Fibo nesnesinin ilk tutturma noktasının koordinatları
   price1=high[70];
   time1=time[70];
//--- Fibo nesnesinin ikinci tutturma noktasının koordinatları
   price2=low[50];
   time2=time[50];

//--- Fibo nesnesinin oluşturulma zamanı
```

```
  bool created=ObjectCreate(0,"Fibo",OBJ_FIBO,0,time1,price1,time2,price2);
  if(created) // Nesne başarılı şekilde oluşturulmuşsa
    {
     //--- Fibo seviyelerinin rengini ayarla
     ObjectSetInteger(0,"Fibo",OBJPROP_LEVELCOLOR,Blue);
     //--- kaç adet Fibo seviyemiz var?
     int levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
     Print("Önceki Fibo seviyeleri = ",levels);
     //---Günlüğe çıktıla => seviye numarası: değer seviye_açıklaması
     for(int i=0;i<levels;i++)
       {
        Print(i,": ",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
              "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
       }
     //--- Birim başına düşen seviye sayısını artırmayı dene
     bool modified=ObjectSetInteger(0,"Fibo",OBJPROP_LEVELS,levels+1);
     if(!modified) // seviye sayısı değiştirilemedi
       {
        Print("Fibo nesnesinin seviyeerinin sayısı değiştirilemedi, hata ",GetLastEr
       }
     //--- sadece bilgilendir
     Print("sonraki Fibo seviyeleri = ",ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS));
     //--- yeni olusturulmuş seviye için bir değer ayarla
     bool added=ObjectSetDouble(0,"Fibo",OBJPROP_LEVELVALUE,levels,133);
     if(added) // seviye için bir değer ayarlandı
       {
        Print("Seviye için, başarılı şekilde bir değer ayarlandı");
        //--- Seviye açıklamasını ayarlamayı da unutma
        ObjectSetString(0,"Fibo",OBJPROP_LEVELTEXT,levels,"my level");
        ChartRedraw(0);
        //--- Fibo nesnesindeki seviyelerin gerçek sayısını al
        levels=ObjectGetInteger(0,"Fibo",OBJPROP_LEVELS);
        Print("Eklemeden sonra Fibo seviyeleri = ",levels);
        //--- emin olmak için, tüm seviyeleri bir kez daha çıktıla
        for(int i=0;i<levels;i++)
          {
           Print(i,":",ObjectGetDouble(0,"Fibo",OBJPROP_LEVELVALUE,i),
                 "  ",ObjectGetString(0,"Fibo",OBJPROP_LEVELTEXT,i));
          }
       }
     else // Fibo nesnesindeki seviyelerin sayısının artırılması denenirse başarısız
       {
        Print("Daha fazla Fibo seviyesi eklenemedi. Hata",GetLastError());
       }
    }
  }
```

Ayrıca Bakınız

Nesne Tipleri, Nesne Özellikleri

ObjectSetInteger

Belirtilen nesne özelliğinin değerini ayarlar. Nesne özelliği datetime, int, color, bool veya char tiplerinde olmalıdır. Fonksiyonun iki çeşidi vardır.

Birinci versiyonda şekillendirici yer almaz

```
bool  ObjectSetInteger(
   long                              chart_id,          // çizelge tanımlayıcı
   string                            name,              // nesne ismi
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,           // özellik tanımlayıcısı
   long                              prop_value         // değer
   );
```

İkinci versiyonda değer, şekillendirici belirtilerek ayarlanır

```
bool  ObjectSetInteger(
   long                              chart_id,          // çizelge tanımlayıcı
   string                            name,              // nesne ismi
   ENUM_OBJECT_PROPERTY_INTEGER      prop_id,           // özellik tanımlayıcısı
   int                               prop_modifier,     // şekillendirici
   long                              prop_value         // değer
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

name

[in] Nesne ismi.

prop_id

[in] Nesne özelliğinin tanıtıcısı. Değer, ENUM_OBJECT_PROPERTY_INTEGER sayımının değerlerinden biri olabilir.

prop_modifier

[in] Belirtilen özelliğin şekillendiricisi. Fibonacci araçlarındaki ve Andrews Dirgenindeki seviye numarasını gösterir. Seviyelerin numaralandırılmasına sıfırdan başlanır.

prop_value

[in] Özelliğin değeri.

Dönüş değeri

Grafiksel nesnenin özelliğini değiştirmek için oluşturulan komut, çizelgeye başarılı şekilde gönderilmişse 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata ile ilgili daha fazla bilgi için GetLastError() fonksiyonunu çağırın.

Not

Fonksiyon, eşzamansız bir çağrı kullanır; bu, işlevin, belirtilen grafiğin sırasına eklenen komutun yürütülmesini beklemediği anlamına gelir. Bunun yerine, hemen kontrol döndürür.

Komut yürütme sonucunu kontrol etmek için, belirtilen nesne mülkiyetini isteyen bir fonksiyon kullanabilirsiniz. Bununla birlikte, bu tür işlevlerin bu çizelgenin sonuna eklendiğini ve yürütme sonucunu beklediklerini ve bu nedenle zaman alıcı olabileceğini unutmamalısınız. Bu özellik, bir grafikte çok sayıda nesneyle çalışırken dikkate alınmalıdır.

Web Renkleri tablosunun oluşturulmasına dair bir örnek

```
//+------------------------------------------------------------------+
//|                                              Table of Web Colors|
//|                        Copyright 2011, MetaQuotes Software Corp |
//|                                        https://www.metaquotes.net |
//+------------------------------------------------------------------+
#define X_SIZE 140      // bir edit nesnesinin genişliği
#define Y_SIZE 33       // bir edit nesnesinin yüksekliği
//+------------------------------------------------------------------+
//| Web renkleri için bir dizi                                       |
//+------------------------------------------------------------------+
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,cli
   clrThistle,clrTomato,clrTurquoise,clrViolet,clrWheat,clrWhite,clrWhiteSmoke,clrYell
  };
//+------------------------------------------------------------------+
//| Bir edit nesnesinin oluşturulup başlatılması                     |
//+------------------------------------------------------------------+
void CreateColorBox(int x,int y,color c)
  {
//--- yeni edit nesnesi için bir isim oluştur
   string name="ColorBox_"+(string)x+"_"+(string)y;
//--- yeni bir edit nesnesi oluştur
   if(!ObjectCreate(0,name,OBJ_EDIT,0,0,0))
     {
      Print("Oluşturulamadı: '",name,"'");
      return;
```

```
      }
//--- genişlik, yükseklik ve koordinatları ayarla
   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);
//--- metin rengini ayara
   if(clrBlack==c) ObjectSetInteger(0,name,OBJPROP_COLOR,clrWhite);
   else            ObjectSetInteger(0,name,OBJPROP_COLOR,clrBlack);
//--- arkaplan rengini ayarla
   ObjectSetInteger(0,name,OBJPROP_BGCOLOR,c);
//--- metni ayarla
   ObjectSetString(0,name,OBJPROP_TEXT,(string)c);
  }
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- 7x20 ölçülerinde, renkli edit nesnelerinden oluşan bir tablo oluştur
   for(uint i=0;i<140;i++)
      CreateColorBox(i%7,i/7,ExtClr[i]);
  }
```

Ayrıca Bakınız

Nesne Tipleri, Nesne Özellikleri

ObjectSetString

Belirtilen nesne özelliğinin değerini ayarlar. Nesne özelliği string tipinde olmalıdır. Fonksiyonun iki çeşidi vardır.

Birinci versiyonda şekillendirici yer almaz

```
bool  ObjectSetString(
   long                          chart_id,          // çizelge tanımlayıcı
   string                        name,              // nesne ismi
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // özellik tanımlayıcısı
   string                        prop_value         // değer
   );
```

İkinci versiyonda değer, şekillendirici belirtilerek ayarlanır

```
bool  ObjectSetString(
   long                          chart_id,          // çizelge tanımlayıcı
   string                        name,              // nesne ismi
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // özellik tanımlayıcısı
   int                           prop_modifier,     // şekillendirici
   string                        prop_value         // değer
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

name

[in] Nesne ismi.

prop_id

[in] Nesne özelliğinin tanıtıcısı. Bu değer, ENUM_OBJECT_PROPERTY_STRING sayımının değerlerinden biri olabilir.

prop_modifier

[in] Belirtilen özelliğin şekillendiricisi. Fibonacci araçlarındaki ve Andrews Dirgenindeki seviye numarasını gösterir. Seviyelerin numaralandırılmasına sıfırdan başlanır.

prop_value

[in] Özelliğin değeri.

Dönüş değeri

Grafiksel nesnenin özelliğini değiştirmek için oluşturulan komut, çizelgeye başarılı şekilde gönderilmişse 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata ile ilgili daha fazla bilgi için GetLastError() fonksiyonunu çağırın.

Not

Fonksiyon, eşzamansız bir çağrı kullanır; bu, işlevin, belirtilen grafiğin sırasına eklenen komutun yürütülmesini beklemediği anlamına gelir. Bunun yerine, hemen kontrol döndürür.

Komut yürütme sonucunu kontrol etmek için, belirtilen nesne mülkiyetini isteyen bir fonksiyon kullanabilirsiniz. Bununla birlikte, bu tür işlevlerin bu çizelgenin sonuna eklendiğini ve yürütme sonucunu beklediklerini ve bu nedenle zaman alıcı olabileceğini unutmamalısınız. Bu özellik, bir grafikte çok sayıda nesneyle çalışırken dikkate alınmalıdır.

Bir nesne yeniden isimlendirildiğinde, aynı anda iki olay ortaya çıkar. Bu olaylar, bir Uzman Danışman veya gösterge içerisinde OnChartEvent() fonksiyonu ile işlenebilir:

- eski isimli nesnenin silinmesi olayı;
- yeni isimli nesnenin oluşturulması olayı.

ObjectGetDouble

Karşılık gelen nesne özelliğinin değerine dönüş yapar. Nesne özelliği double tipinde olmalıdır. Fonksiyonun iki çeşidi vardır.

1. Hemen, özellik değerine dönüş yapar.

```
double  ObjectGetDouble(
   long                          chart_id,          // çizelge tanımlayıcı
   string                        name,              // nesne ismi
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // özellik tanımlayıcı
   int                           prop_modifier=0    // istenmesi durumunda, özellik
   );
```

2. Fonksiyonun başarı durumuna göre, 'true' veya 'false' değerine dönüş yapar. Başarılı sonuç mevcutsa, son parametreye referansla geçirilen bir değişkene yerleştirilir.

```
bool  ObjectGetDouble(
   long                          chart_id,          // çizelge tanımlayıcı
   string                        name,              // nesne ismi
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,           // özellik tanımlayıcı
   int                           prop_modifier,     // özellik şekillendirici
   double&                       double_var         // özellik değerini burada kabul
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

name

[in] Nesne ismi.

prop_id

[in] Nesne özelliğinin tanıtıcısı. Bu değer ENUM_OBJECT_PROPERTY_DOUBLE sayımının değerlerinden biri olabilir.

prop_modifier

[in] Belirtilen özelliğin şekillendiricisi. İlk versiyonda, varsayılan şekillendirici değeri sıfıra eşittir. Çoğu özellik için bir şekillendiriciye gerek duyulmaz. Fibonacci araçlarındaki ve Andrews Dirgenindeki seviye numarasını gösterir. Seviyelerin numaralandırılmasına sıfırdan başlanır.

double_var

[out] İstenen özellik değerini alacak olan double tipli değişken.

Dönüş değeri

İlk çağrı versiyonu için, double tipli bir değer.

İkinci durumda, eğer özellik mevcutsa ve değer double_var değişkenine yerleştirilmişse 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata ile ilgili daha fazla bilgi için GetLastError() fonksiyonunu çağırın.

Not

Fonksiyon, eşzamanlı bir çağrı kullanır; bu, fonksiyonun çağrı öncesinde bu grafik için yerine getirilmiş tüm komutların yürütülmesini beklediği anlamına gelir, bu nedenle bu fonksiyon çok zaman alıcı olabilir. Bu özellik, bir grafikte çok sayıda nesneyle çalışırken dikkate alınmalıdır.

ObjectGetInteger

Karşılık gelen nesne özelliğinin değerine dönüş yapar. Nesne özelliği datetime, int, color, bool veya char tiplerinde olmalıdır. Fonksiyonun iki çeşidi vardır.

1. Hemen, özellik değerine dönüş yapar.

```
long  ObjectGetInteger(
   long                          chart_id,          // çizelge tanımlayıcı
   string                        name,              // nesne ismi
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,           // özellik tanımlayıcı
   int                           prop_modifier=0    // istenmesi durumunda özellik
   );
```

2. Fonksiyonun başarı durumuna göre, 'true' veya 'false' değerine dönüş yapar. Başarılı sonuç mevcutsa, son parametreye referansla geçirilen bir değişkene yerleştirilir.

```
bool  ObjectGetInteger(
   long                          chart_id,          // çizelge tanımlayıcı
   string                        name,              // nesne ismi
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,           // özellik tanımlayıcı
   int                           prop_modifier,     // özellik şekillendirici
   long&                         long_var           // özelik değerini burada farz
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

name

[in] Nesne ismi.

prop_id

[in] Nesne özelliğinin tanıtıcısı. Değer, ENUM_OBJECT_PROPERTY_INTEGER sayımının değerlerinden biri olabilir.

prop_modifier

[in] Belirtilen özelliğin şekillendiricisi. İlk versiyonda, varsayılan şekillendirici değeri sıfıra eşittir. Çoğu özellik için bir şekillendiriciye gerek duyulmaz. Fibonacci araçlarındaki ve Andrews Dirgenindeki seviye numarasını gösterir. Seviyelerin numaralandırılmasına sıfırdan başlanır.

long_var

[out] İstenen özellik değerini alacak olan long tipli değişken.

Dönüş değeri

İlk çağrı versiyonu için, long tipli bir değer.

İkinci durumda, eğer özellik mevcutsa ve değer long_var değişkenine yerleştirilmişse 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata ile ilgili daha fazla bilgi için GetLastError() fonksiyonunu çağırın.

Not

Fonksiyon, eşzamanlı bir çağrı kullanır; bu, fonksiyonun çağrı öncesinde bu grafik için yerine getirilmiş tüm komutların yürütülmesini beklediği anlamına gelir, bu nedenle bu fonksiyon çok zaman alıcı olabilir. Bu özellik, bir grafikte çok sayıda nesneyle çalışırken dikkate alınmalıdır.

ObjectGetString

Karşılık gelen nesne özelliğinin değerine dönüş yapar. Nesne özelliği string tipinde olmalıdır. Fonksiyonun iki çeşidi vardır.

1. Hemen, özellik değerine dönüş yapar.

```
string  ObjectGetString(
   long                          chart_id,          // çizelge tanımlayıcı
   string                        name,              // nesne ismi
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // özellik tanımlayıcı
   int                           prop_modifier=0    // stenmesi durumunda, özellik s
   );
```

2. Fonksiyonun başarı durumuna göre, 'true' veya 'false' değerine dönüş yapar. Başarılı sonuç mevcutsa, son parametreye referansla geçirilen bir değişkene yerleştirilir.

```
bool  ObjectGetString(
   long                          chart_id,          // çizelge tanımlayıcı
   string                        name,              // nesne ismi
   ENUM_OBJECT_PROPERTY_STRING   prop_id,           // özellik tanımlayıcı
   int                           prop_modifier,     // özellik şekillendirici
   string&                       string_var         // özelik değerini burada farz e
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

name

[in] Nesne ismi.

prop_id

[in] Nesne özelliğinin tanıtıcısı. Bu değer, ENUM_OBJECT_PROPERTY_STRING sayımının değerlerinden biri olabilir.

prop_modifier

[in] Belirtilen özelliğin şekillendiricisi. İlk versiyonda, varsayılan şekillendirici değeri sıfıra eşittir. Çoğu özellik için bir şekillendiriciye gerek duyulmaz. Fibonacci araçlarındaki ve Andrews Dirgenindeki seviye numarasını gösterir. Seviyelerin numaralandırılmasına sıfırdan başlanır.

string_var

[out] İstenen özellik değerini alacak olan string tipli değişken.

Dönüş değeri

İlk çağrı versiyonu için string tipli bir değer.

İkinci durumda, eğer özellik mevcutsa ve değer string_var değişkenine yerleştirilmişse 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata ile ilgili daha fazla bilgi için GetLastError() fonksiyonunu çağırın.

Not

Fonksiyon, eşzamanlı bir çağrı kullanır; bu, fonksiyonun çağrı öncesinde bu grafik için yerine getirilmiş tüm komutların yürütülmesini beklediği anlamına gelir, bu nedenle bu fonksiyon çok zaman alıcı olabilir. Bu özellik, bir grafikte çok sayıda nesneyle çalışırken dikkate alınmalıdır.

Bir nesne yeniden isimlendirildiğinde, aynı anda iki olay ortaya çıkar. Bu olaylar, bir Uzman Danışman veya gösterge içerisinde OnChartEvent() fonksiyonu ile işlenebilir:

- eski isimli nesnenin silinmesi olayı;
- yeni isimli nesnenin oluşturulması olayı.

TextSetFont

Bu fonksiyon, çizim yöntemlerini kullanarak görüntülenen yazı tipini ayarlar ve işlemin sonucuna dönüş yapar. Varsayılan yazı tipi olarak, Arial -120 (12 pt) kullanılmaktadır.

```
bool  TextSetFont(
   const string  name,            // yazı tipinin ismi veya yazı tipi dosyasının disk
   int           size,            // yazı tipi boyutu
   uint          flags,           // bayrakların kombinasyonu
   int           orientation=0    // metnin eğim açısı
   );
```

Parametreler

name

[in] Sistemdeki yazı tipi ismi veya yazı tipini içeren kaynağın ismi veya yazı tipi dosyasının disk üzerindeki konumu.

size

[in] Negatif veya pozitif sayılarla ayarlanabilir olan yazı tipi boyutu. Bir pozitif değerin kullanılması durumunda, görüntülenen metnin boyutu sistemde ayarlı yazı tipi boyutuna bağımlıdır. Bir negatif değerin kullanılması durumunda ise, değer on noktalık birimlerle ayarlanır ve metin boyutu işletim sistemi ayarlarına bağlıdır ("standart ölçek" veya "büyük ölçek"). Kullanılan modların farkları hakkında daha fazla bilgi alabilmek amacıyla aşağıdaki not kısmına bakınız.

bayraklar

[in] Yazı tipi stilini belirleyen bayrakların kombinasyonu.

orientation

[in] Metnin X eksenine göre eğimi - ölçü birimi 0.1 derecedir. Yani *orientation=450* değeri, 45 derecelik eğim açısı anlamına gelir.

Dönüş değeri

Yazı tipi başarılı şekilde yüklenmişse 'true', aksi durumda ise 'false' dönüşü yapar. Muhtemel hata kodları:

- ERR_INVALID_PARAMETER(4003) - *name* değişkeni, NULL veya "" (boş metin) ifade ediyor
- ERR_INTERNAL_ERROR(4001) - işletim sistemi hatası (örneğin, olmayan bir yazı tipinin ayarlanmaya çalışılması).

Not

Yazı tipinin isminde "::" kullanılmışsa, yazı tipi EX5 kaynağından indirilir. Eğer yazı tipi ismi *name* bir uzantı ile belirtilmişse, yazı tipi ilgili dosyadan yüklenir; dosya adresi "\" veya "/" ile başlıyorsa, dosya MQL5 dizininde aranır. Aksi durumda, TextSetFont() fonksiyonunu çağıran EX5 dosyasının konumuna göre arama yapılır.

Yazı tipi boyutu pozitif veya negatif değerler ile ayarlanır. Bu, metin boyutunun işletim sistemi ayarlarına (boyut ölçeğine) olan bağımlılığını belirler.

- Boyut, bir pozitif sayı ile belirlenmişse, mantıksal ölçü birimlerinden aygıtın fiziksel ölçü birimlerine (pikseller) dönüşüm gerçekleştirilir ve dönüşüm gerçekleştirildiği zaman bu boyut, mevcut yazı tiplerinden alınan sembol gliflerinin yüksekliğine karşılık gelir. Eğer TextOut()

fonksiyonu ile görüntülenen metinler ve OBJ_LABEL ("Label") grafiksel nesnesi ile görüntülenen metinler çizelge üzerinde birlikte kullanılacaksa bu yöntem önerilmez.

- Eğer boyut, bir negatif sayı ile belirlenmişse, bu sayının, onlu mantıksal noktalarla ayarlanmış olması beklenir; yani verilen sayı ona bölünür (-350 sayısı 35 mantıksal noktaya karşılık gelir). Alınan değer, mantıksal ölçü birimlerinden aygıtın fiziksel ölçü birimlerine (pikseller) dönüştürülür ve mevcut yazı tiplerinden alınan sembolün mutlak yüksek değerine karşılık gelir. Ekranda yer alan bir metnin boyutunu OBJ_LABEL nesnesindeki gibi ayarlamak için, nesne özelliklerinde belirtilen yazı tipi boyutunu -10 ile çarpın.

Bayraklar, stil bayrakları ve yazı tipi genişliğini belirleyen bir bayrakla birlikte kombinasyon şeklinde kullanılabilir. Bayrakların isimleri aşağıda gösterilmiştir.

Yazı tipi stilini belirleyen bayraklar

Bayrak	Açıklama
FONT_ITALIC	İtalik
FONT_UNDERLINE	Alt çizgi
FONT_STRIKEOUT	Üstü çizili

Yazı tipi genişliğini belirleyen bayraklar

Bayrak
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

Ayrıca Bakınız

Kaynaklar, ResourceCreate(), ResourceSave(), TextOut()

TextOut

Bir metni özel bir dizi (tampon) içerisinde görüntüler ve işlemin sonucuna dönüş yapar. Söz konusu dizi, grafiksel kaynak oluşturma amacıyla tasarlanır.

```
bool  TextOut(
   const string       text,          // görüntülenen metin
   int                x,             // X koordinatı
   int                y,             // Y koordinatı
   uint               anchor,        // tutturma (çapa) tipi
   uint               &data[],       // çıktı tamponu
   uint               width,         // piksel bazında tampon genişliği
   uint               height,        // piksel bazında tampon yüksekliği
   uint               color,         // metin rengi
   ENUM_COLOR_FORMAT  color_format   // çıktı için renk biçimi
   );
```

Parametreler

text

[in] Tampona yazılıp görüntülenecek metin. Sadece tek satırlık metin görüntülenir.

x

[in] Görüntülenecek metnin tutturma noktasının X koordinatı.

y

[in] Görüntülenecek metnin tutturma noktasının Y koordinatı.

anchor

[in] Görüntülenecek metnin tutturma noktası için 9 ön-tanımlı yöntemden birinin değeri. Bu değer, iki bayraklı bir kombinasyon ile belirlenir - yatay ve dikey metin hizalama bayrakları. Söz konusu bayraklar, Not kısmında aşağıda listelenmiştir.

data[]

[in] Metnin görüntüleneceği tampon. Bu tampon, grafiksel kaynak oluşturma amacıyla kullanılır.

genişlik

[in] Piksel bazında tampon genişliği.

height

[in] Piksel bazında tampon yüksekliği.

color

[in] Metin rengi.

color_format

[in] Renk değeri, ENUM_COLOR_FORMAT sayımı değerleri ile ayarlanır.

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Not

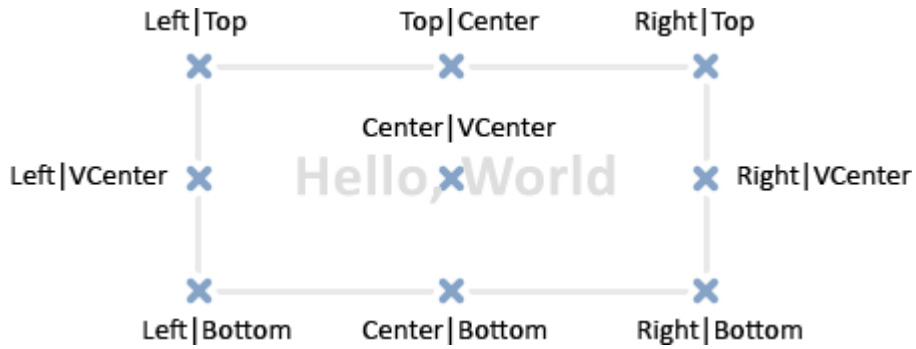
tutturma noktası (çapa) koordinatları, yatay ve dikey metin hizalama bayraklarından oluşan ikili bir kombinasyon ile belirlenir. Yatay metin hizalama bayrakları:

- TA_LEFT - karakter kutusunun sol tarafına yerleştirilen tutturma noktası
- TA_CENTER - karakter kutusunun ortasına yerleştirilen tutturma noktası
- TA_RIGHT - karakter kutusunun sağ tarafına yerleştirilen tutturma noktası

Dikey metin hizalama bayrakları:

- TA_TOP - karakter kutusunun üst tarafına yerleştirilen tutturma noktası
- TA_VCENTER - karakter kutusunun ortasına yerleştirilen tutturma noktası
- TA_BOTTOM - karakter kutusunun alt tarafına yerleştirilen tutturma noktası

Muhtemel bayrak kombinasyonları ve seçilen tutturma noktaları resimde gösterilmektedir.



Örnek:

```
//--- çizimde kullanılacak tuvalin yükseklik ve genişliği
#define IMG_WIDTH  200
#define IMG_HEIGHT 200
//--- betiği çalıştırmadan önce parametreler penceresini göster
#property script_show_inputs
//--- renk biçimi ayarlama modunu etkinleştir
input ENUM_COLOR_FORMAT clr_format=COLOR_FORMAT_XRGB_NOALPHA;
//--- tamponun çizimi
uint ExtImg[IMG_WIDTH*IMG_HEIGHT];
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- çizim için OBJ_BITMAP_LABEL nesnesini oluştur
   ObjectCreate(0,"CLOCK",OBJ_BITMAP_LABEL,0,0,0);
//--- CLOCK nesnesinin yazımında kullanılacak grafiksel kaynağın ismini belirle
   ObjectSetString(0,"CLOCK",OBJPROP_BMPFILE,"::IMG");

//--- ek değişkenler
   double a;            // ok köşesi
   uint   nm=2700;      // dakika köşesi
   uint   nh=2700*12;   // saat sayacı
   uint   w,h;          // metin boyutunu almak için bir değişken
```

```
   int     x,y;           // mevcut metin koordinatlarının hesaplanması için değişkenler

//--- script sonlana kadar, sonsuz bir döngü kullanarak saat ibrelerini çevir
   while(!IsStopped())
     {
      //--- saat çizim tamponunun dizisini temizle
      ArrayFill(ExtImg,0,IMG_WIDTH*IMG_HEIGHT,0);
      //--- saat kadranına rakamlarını çizmek için yazı tipini ayarla
      TextSetFont("Arial",-200,FW_EXTRABOLD,0);
      //--- saat kadranını çiz
      for(int i=1;i<=12;i++)
        {
         //--- akrebin kadrandaki mevcut boyutunu al
         TextGetSize(string(i),w,h);
         //--- akrebin kadran üzerindeki mevcut koordinatlarını hesapla
         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);
         //--- akrebi (saati) ExtImg[] tamponuna çıktıla
         TextOut(string(i),x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,
        }
      //--- şimdi dakika ibresi (yelkovan) çizmek için bir yazı tipi belirle
      TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nm%3600));
      //--- yelkovan boyutunu al
      TextGetSize("----->",w,h);
      //--- yelkovanın kadran üzerindeki konumunu al
      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);
      //--- yelkovanı (dakikayı) ExtImg[] tamponuna çıktıla
      TextOut("----->",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_f

      //--- şimdi, yelkovanı çizmek için yazı tipi ayarla
      TextSetFont("Arial",-200,FW_EXTRABOLD,-int(nh/12%3600));
      TextGetSize("==>",w,h);
      //--- akrebin kadran üzerindeki koordinatlarını hesapla
      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);
      //--- akrebi (saati) ExtImg[] tamponuna çıktıla
      TextOut("==>",x,y,TA_LEFT|TA_TOP,ExtImg,IMG_WIDTH,IMG_HEIGHT,0xFFFFFFFF,clr_form

      //--- grafiksel kaynağı güncelle
      ResourceCreate("::IMG",ExtImg,IMG_WIDTH,IMG_HEIGHT,0,0,IMG_WIDTH,clr_format);
      //--- çizelgeyi zorla güncelle
      ChartRedraw();

      //--- dakika ve saat sayaçlarını artır
      nm+=60;
      nh+=60;
```

```
      //--- sayacın artırılması için kısa bir duraklama
      Sleep(10);
     }
//--- script işlemi tamamlandığında CLOCK nesnesini sil
   ObjectDelete(0,"CLOCK");
//---
  }
```

Ayrıca Bakınız

Kaynaklar, ResourceCreate(), ResourceSave(), TextGetSize(), TextSetFont()

TextGetSize

Mevcut yazı tipi ayarları için çizgi genişliğine ve yüksekliğine dönüş yapar.

```
bool  TextGetSize(
   const string       text,          // metin dizgisi
   uint&               width,        // piksel bazında tampon genişliği
   uint&               height        // piksel bazında tampon yüksekliği
   );
```

Parametreler

text

[in] Genişlik ve yüksekliğin alınacağı dizgi.

genişlik

[out] Genişliğin alınması için giriş parametresi.

height

[out] Yüksekliğin alınması için giriş parametresi.

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Muhtemel hata kodları:

- ERR_INTERNAL_ERROR(4001) - işletim sistemi hatası.

Ayrıca Bakınız

Kaynaklar, ResourceCreate(), ResourceSave(), TextSetFont(), TextOut()

Teknik Gösterge Fonksiyonları

iMA, iAC, iMACD, iIchimoku vb. gibi tüm fonksiyonlar, müşteri terminalinin global önbelleğinde karşılık gelen göstergenin bir kopyasını oluştururlar. Aynı parametrelere sahip bir gösterge kopyası zaten mevcutsa yeni kopya oluşturulmaz ve mevcut kopyanın referans sayacı artar.

Bu fonksiyonlar, göstergenin uygun kopyasının tanıtıcı değerine dönüş yapar. Bu değer kullanılarak, karşılık gelen göstergeye dair veriler alınabilir. Karşılık gelen tampon verisi (teknik göstergeler, göstergeye bağlı olarak, 1 ila 5 arasında değişebilen tamponlarda hesaplanan veriler içerir) CopyBuffer() fonksiyonu kullanılarak bir MQL5 programına kopyalanabilir.

Göstergenin oluşturulmasıyla birlikte, gösterge verisini hemen kullanamazsınız; gösterge verilerinin hesaplanması zaman alacaktır. Bu yüzden, gösterge tanıtıcı değerlerinin OnInit() içerisinde oluşturulması daha iyidir. iCustom() fonksiyonu, karşılık gelen özel göstergeyi oluşturur ve başarılı olması durumunda onun tanıtıcı değerine dönüş yapar. Özel göstergeler 512 adete kadar gösterge tamponu içerebilirler. Tamponların içerikleri, elde edilen tanıtıcı değer kullanılarak, CopyBuffer() fonksiyonu ile elde edilebilir.

Bir teknik göstergeyi oluşturmak için evrensel bir yöntem, IndicatorCreate() fonksiyonunun kullanımıdır. Fonksiyon, şu verileri giriş parametresi olarak kabul eder:

- Sembol ismi.
- Zaman aralığı
- Oluşturulacak göstergenin tipi.
- Göstergenin giriş parametrelerinin sayısı.
- tüm gerekli giriş parametrelerini içeren MqlParam tipi bir dizi.

Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir; bunun için, gösterge tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Not Bir MQL5 programı içerisinde, bir gösterge fonksiyonunun aynı parametrelerle tekrar çağrılması, referans sayacının birden fazla artırılmasına neden olmaz; sayaç sadece 1 ile artırılacaktır. Buna rağmen, gösterge tanıtıcı değerlerinin OnInit() veya sınıf yapıcısı içine yazılmaları ve daha sonra diğer fonksiyonlarla kullanılması tavsiye edilir. MQL5 programı sonlandırıldığında, referans sayacı azalır.

Tüm gösterge fonksiyonları en az iki parametreye sahiptir - sembol ve periyot. Sembol için kullanılan NULL değeri geçerli sembol anlamına gelir ve 0 periyot değeri ile geçerli zaman aralığı kast edilir.

Fonksiyon	Göstergenin tanıtıcı değerine dönüş yapar:
iAC	Accelerator Oscillator
iAD	Accumulation/Distribution
iADX	Average Directional Index
iADXWilder	Average Directional Index by Welles Wilder
iAlligator	Alligator
iAMA	Adaptive Moving Average
iAO	Awesome Oscillator

Fonksiyon	Göstergenin tanıtıcı değerine dönüş yapar:
iATR	Average True Range
iBearsPower	Bears Power
iBands	Bollinger Bands®
iBullsPower	Bulls Power
iCCI	Commodity Channel Index
iChaikin	Chaikin Oscillator
iCustom	Özel gösterge
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
iWPR	Williams' Percent Range

Fonksiyon	Göstergenin tanıtıcı değerine dönüş yapar:
iVIDyA	Variable Index Dynamic Average
iVolumes	Volumes

iAC

Accelerator Oscillator göstergesini terminalin global bellek havuzunda oluşturur ve gösterge tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iAC(
   string           symbol,     // sembol ismi
   ENUM_TIMEFRAMES  period      // periyot
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri ENUM_TIMEFRAMES sayımının değerlerinden biri olabilir, 0 değeri, mevcut zaman aralığını gösterir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAC.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir"
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- iAC grafiği
#property indicator_label1  "iAC"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
```

```
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAC,               // iAC kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iAC;          // fonksiyon tipi
input string                symbol=" ";             // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // zaman aralığı
//--- gösterge tamponları
double iACBuffer[];
double iACColors[];
//--- iAC göstergesinin tanıtıcı değerini depolamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Accelerator Oscillator göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iACBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iACColors,INDICATOR_COLOR_INDEX);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iAC)
      handle=iAC(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AC);
//--- işleyici oluşturulmadıysa
```

```
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iAC göstergesinin tanıtıcı değeri, %s/%s sembolü için alınamadı, ha
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Accelerator Oscillator göstergesinin hesaplandığı sembol ve zaman aralığı değer
   short_name=StringFormat("iAC(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iAC göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu gösterge değerlerinin ilk hesaplanması ise veya iAC gösterge değerleri değiş
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iACBuffer dizisinin büyüklüğü iAC gösterges
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
```

```
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iACBuffer ve iACColors dizilerini Accelerator Oscillator göstergesinin değerleri
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArraysFromBuffer(iACBuffer,iACColors,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Accelerator Oscillator göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iAC göstergesi ile, gösterge tamponlarının doldurulması          |
//+------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],        // Accelerator Oscillator değerleri
                          double &color_indexes[], // renk tamponu (renk indislerini d
                          int ind_handle,          // iAC göstegesinin işleyicisi
                          int amount               // kopyalanan değerlerin sayısı
                          )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iACBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iAC göstergesinden veriler kopyalanamadı, hata kodu %d",GetLastErro
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- renk indislerini kopyala
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iAC göstergesinden veriler kopyalanamadı, hata kodu %d",GetLastErro
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
```

```
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iAD

Accumulation/Distribution göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iAD(
   string               symbol,             // sembol ismi
   ENUM_TIMEFRAMES      period,             // periyot
   ENUM_APPLIED_VOLUME  applied_volume      // hesaplamada kullanılacak hacim tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri ENUM_TIMEFRAMES sayımının değerlerinden biri olabilir, 0 değeri, mevcut zaman aralığını gösterir.

applied_volume

[in] Kullanılan hacim tipi. ENUM_APPLIED_VOLUME değerlerinden biri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                    Demo_iAD.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "değerlerin nasıl alınacağını gösterir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iAD grafiği
#property indicator_label1  "iAD"
#property indicator_type1   DRAW_LINE
```

```
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAD,               // iAD kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iAD;          // fonksiyon tipi
input ENUM_APPLIED_VOLUME   volumes;                // kullanılan hacim
input string                symbol=" ";             // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // zaman aralığı
//--- gösterge tamponu
double          iADBuffer[];
//--- iAD göstergesinin tanıtıcı değerini depolamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Accumulation/Distribution göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iADBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iAD)
      handle=iAD(name,period,volumes);
   else
     {
```

```
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=volumes;
      handle=IndicatorCreate(name,period,IND_AD,1,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iAD göstergesinin tanıtıcı değeri, %s/%s sembolü için alınamadı, ha
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Accumulation/Distribution göstergesinin hesaplandığı sembol ve zaman aralığı değ
   short_name=StringFormat("iAD(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iAD göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu gösterge değerlerinin ilk hesaplanması ise veya iAD gösterge değerleri değişt
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

```
     {
      //--- verilen sembol ve periyot için iADBuffer dizisinin büyüklüğü iAD gösterges
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iADBuffer dizisini Accumulation/Distribution göstergesinin değerleri ile doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iADBuffer,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Accumulation/Distribution göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iAD göstergesi ile, gösterge tamponlarının doldurulması          |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // Accumulation/Distribution çizgisi için
                         int ind_handle,     // iAD göstegesinin işleyicisi
                         int amount          // kopyalanmış değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iADBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iAD göstergesinden veriler kopyalanamadı, hata kodu %d",GetLastErro
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
```

```
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iADX

Average Directional Movement Index göstergesinin tanıtıcı değerine dönüş yapar.

```
int  iADX(
   string           symbol,          // sembol ismi
   ENUM_TIMEFRAMES  period,          // peryiot
   int              adx_period       // ortalama periyodu
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

adx_period

[in] Endeks değerinin hesaplanacağı periyot.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Not

Tampon numaraları şu şekildedir: 0 - MAIN_LINE, 1 - PLUSDI_LINE, 2 - MINUSDI_LINE.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iADX.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_separate_window
#property indicator_buffers 3
```

```
#property indicator_plots   3
//--- ADX grafiği
#property indicator_label1  "ADX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- DI_plus grafiği
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- DI_minus grafiği
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iADX,              // iADX kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iADX;         // fonksiyon tipi
input int                   adx_period=14;          // hesaplama periyodu
input string                symbol=" ";             // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // zaman aralığı
//--- gösterge tamponları
double          ADXBuffer[];
double          DI_plusBuffer[];
double          DI_minusBuffer[];
//--- iADX göstergesinin tanıtıcı değeri için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Average Directional Movement Index göstergesindeki değerlerin sayısını tutacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
```

```
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iADX)
      handle=iADX(name,period,adx_period);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADX,1,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("%s/%s sembolü için iADX göstergesinin tanıtıcı değeri oluşturulamac
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Average Directional Movement Index göstergesinin hesaplandığı sembol ve zaman a
   short_name=StringFormat("iADX(%s/%s period=%d)",name,EnumToString(period),adx_perio
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iADX göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu gösterge değerlerinin ilk hesaplanması ise veya iADX gösterge değerleri değiş
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iADXBuffer dizisinin büyüklüğü iAD gösterge
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- Diziyi, Average Directional Movement Index göstergesinin değerleri ile doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_c
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Average Directional Movement Index göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iADX göstergesi ile, gösterge tamponlarının doldurulması         |
```

```
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &adx_values[],      // ADX çizgisinin gösterge tampo
                           double &DIplus_values[],   // DI+ için gösterge tamponu
                           double &DIminus_values[],  // DI- için gösterge tamponu
                           int ind_handle,            // iADXWilder göstergesinin tanı
                           int amount                 // kopyalanan değerler
                           )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iADXBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iADX göstergesinden veriler kopyalanamadı, hata kodu %d",GetLastErr
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }

//--- DI_plusBuffer dizisinin bir kısmını 1 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iADX göstergesinden veriler kopyalanamadı, hata kodu %d",GetLastErr
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }

//--- DI_plusBuffer dizisinin bir kısmını 2 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iADX göstergesinden veriler kopyalanamadı, hata kodu %d",GetLastErr
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iADXWilder

Average Directional Movement Index by Welles Wilder göstergesinin tanıtıcı değerine dönüş yapar.

```
int  iADXWilder(
   string           symbol,          // sembol ismi
   ENUM_TIMEFRAMES  period,          // peryiot
   int              adx_period       // ortalama periyodu
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

adx_period

[in] Endeks değerinin hesaplanacağı periyot.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Not

Tampon numaraları şu şekildedir: 0 - MAIN_LINE, 1 - PLUSDI_LINE, 2 - MINUSDI_LINE.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                   iADXWilder.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_separate_window
#property indicator_buffers 3
```

```
#property indicator_plots   3
//--- ADX grafiği
#property indicator_label1  "ADX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- DI_plus grafiği
#property indicator_label2  "DI_plus"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrYellowGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- DI_minus grafiği
#property indicator_label3  "DI_minus"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrWheat
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iADXWilder,         // iADXWilder kullan
   Call_IndicatorCreate     // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iADXWilder;   // fonksiyon tipi
input int                   adx_period=14;          // hesaplama periyodu
input string                symbol=" ";             // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // zaman aralığı
//--- gösterge tamponları
double          ADXBuffer[];
double          DI_plusBuffer[];
double          DI_minusBuffer[];
//--- iADXWilder göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Average Directional Movement Index by Welles Wilder göstergesindeki değerlerin s
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
```

```
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,ADXBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,DI_plusBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,DI_minusBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iADXWilder)
      handle=iADXWilder(name,period,adx_period);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=adx_period;
      handle=IndicatorCreate(name,period,IND_ADXW,1,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iADXWilder göstergesinin tanıtıcı değeri, %s/%s sembolü için oluştu
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Average Directional Movement Index by Welles Wilder göstergesinin hesaplandığı s
   short_name=StringFormat("iADXWilder(%s/%s period=%d)",name,EnumToString(period),adx
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iADXWilder göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iADXWilder göstergesinin değ
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iADXBuffer dizisinin büyüklüğü iADXWilder g
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- Average Directional Movement Index by Welles Wilder göstergesinin değerleri ile
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArraysFromBuffers(ADXBuffer,DI_plusBuffer,DI_minusBuffer,handle,values_to_c
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Average Directional Movement Index göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iADXWilder göstergesi ile, gösterge tamponlarının doldurulması   |
```

```
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &adx_values[],      // ADX çizgisinin gösterge tampo
                           double &DIplus_values[],   // DI+ için gösterge tamponu
                           double &DIminus_values[],  // DI- için gösterge tamponu
                           int ind_handle,            // iADXWilder göstergesinin tanı
                           int amount                 // kopyalanan değerler
                           )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iADXBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,adx_values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iADXWilder göstergesinin verileri kopyalanamadı, hata kodu %d",GetL
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }

//--- DI_plusBuffer dizisinin bir kısmını 1 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,1,0,amount,DIplus_values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iADXWilder göstergesinin verileri kopyalanamadı, hata kodu %d",GetL
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }

//--- DI_plusBuffer dizisinin bir kısmını 2 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,2,0,amount,DIminus_values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iADXWilder göstergesinin verileri kopyalanamadı, hata kodu %d",GetL
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iAlligator

Fonksiyon, Alligator (timsah) göstergesinin tanıtıcı değerine dönüş yapar.

```
int  iAlligator(
   string               symbol,            // sembol ismi
   ENUM_TIMEFRAMES      period,            // periyot
   int                  jaw_period,        // çenelerin hesaplanması için periyot
   int                  jaw_shift,         // çenelerin yatay kaydırma değeri
   int                  teeth_period,      // dişlerin hesaplanması için periyot
   int                  teeth_shift,       // dişlerin yatay kaydırma değeri
   int                  lips_period,       // dudakların hesaplanması için periyot
   int                  lips_shift,        // dudakların yatay kaydırma değeri
   ENUM_MA_METHOD       ma_method,         // düzleştirme tipi
   ENUM_APPLIED_PRICE   applied_price      // fiyat veya işleyici tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

jaw_period

[in] Mavi çizgi için ortalama periyodu (Timsahın dişleri)

jaw_shift

[in] fiyat çizelgesine göre mavi çizginin kaydırma değeri.

teeth_period

[in] Kırmızı çizgi için ortalama periyodu (Timsahın dişleri).

teeth_shift

[in] fiyat çizelgesine göre kırmızı çizginin kaydırma değeri.

lips_period

[in] Yeşil çizgi için ortalama periyodu (Timsahın dudakları).

lips_shift

[in] fiyat çizelgesine göre yeşil çizginin kaydırma değeri.

ma_method

[in] Ortalama yöntemi. ENUM_MA_METHOD değerlerinden biri olabilir.

applied_price

[in] Kullanılan fiyat. ENUM_APPLIED_PRICE sayımındaki fiyat sabitlerinden biri veya başka bir göstergenin tanıtıcı değeri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Not

Tampon numaraları şu şekildedir: 0 - GATORJAW_LINE, 1 - GATORTEETH_LINE, 2 - GATORLIPS_LINE.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                Demo_iAlligator.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla
#property description "Diğer tüm parametreler Alligator'de olduğu gibidir."

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- Jaws grafiği
#property indicator_label1  "Jaws"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Teeth grafiği
#property indicator_label2  "Teeth"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- Lips grafiği
#property indicator_label3  "Lips"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrLime
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
```

```
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAlligator,         // iAlligator kullan
   Call_IndicatorCreate     // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iAlligator;   // fonksiyon tipi
input string                symbol=" ";             // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // zaman aralığı
input int                   jaw_period=13;          // Jaw (çene) çizgisinin periyodu
input int                   jaw_shift=8;            // Jaw çizgisinin kaydırma değeri
input int                   teeth_period=8;         // Teeth (diş) çizgisinin periyodu
input int                   teeth_shift=5;          // Teeth çizgisinin kaydırma değeri
input int                   lips_period=5;          // Lips (dudak) çizgisinin periyodu
input int                   lips_shift=3;           // Lips çizgisinin kaydırma değeri
input ENUM_MA_METHOD        MA_method=MODE_SMMA;    // Alligator çizgilerinin ortalama
input ENUM_APPLIED_PRICE    applied_price=PRICE_MEDIAN;// Alligator hesaplamasında kul
//--- gösterge tamponları
double          JawsBuffer[];
double          TeethBuffer[];
double          LipsBuffer[];
//--- iAlligator göstergesinin tanıtıcı değerini depolamak için bir değişken
int     handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Alligator göstergesindeki değerlerin sayısını koruyacağız
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,JawsBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,TeethBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,LipsBuffer,INDICATOR_DATA);
//--- çizgilerin kaydırma değerlerini ayarla
   PlotIndexSetInteger(0,PLOT_SHIFT,jaw_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,teeth_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,lips_shift);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
```

```
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   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
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[8];
     //--- Alligator çizgilerinin periyotları ve kaydırma değerleri
      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;
//--- düzleştirme tipi
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
//--- fiyat tipi
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
//--- işleyiciyi oluştur
      handle=IndicatorCreate(name,period,IND_ALLIGATOR,8,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iAlligator göstergesinin tanıtıcı değeri, %s/%s sembolü için alınar
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Alligator göstergesinin hesaplandığı sembol ve zaman aralığı değerlerini göster
```

```
   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);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iAlligator göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu gösterge değerlerinin ilk hesaplanması ise veya iAlligator gösterge değerleri
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için JawsBuffer dizisinin büyüklüğü iAlligator g
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- dizileri Alligator göstergesinin değerleriyle doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArraysFromBuffers(JawsBuffer,jaw_shift,TeethBuffer,teeth_shift,LipsBuffer,l
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
```

```
                              TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                              short_name,
                              values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Alligator göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iAlligator göstergesi ile, gösterge tamponlarının doldurulması   |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &jaws_buffer[],  // Jaw çizgisi için gösterge tampor
                           int j_shift,            // Jaw (çene) çizgisinin kaydırma d
                           double &teeth_buffer[], // Teeth (diş) çiz
                           int t_shift,            // Teeth çizgisinin kaydırma değeri
                           double &lips_buffer[],  // Lips çizgisi için gösterge tampo
                           int l_shift,            // Lips çizgisinin kaydırma değeri
                           int ind_handle,         // iAlligator göstergesinin tanıtıc
                           int amount              // kopyalanan değerlerin sayısı
                           )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- JawsBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,-j_shift,amount,jaws_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iAlligator göstergesinin değerleri kopyalanamadı, hata kodu %d",Get
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }

//--- TeethBuffer dizisinin bir kısmını 1 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,1,-t_shift,amount,teeth_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iAlligator göstergesinin değerleri kopyalanamadı, hata kodu %d",Get
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }

//--- LipsBuffer dizisinin bir kısmını 2 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,2,-l_shift,amount,lips_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iAlligator göstergesinin değerleri kopyalanamadı, hata kodu %d",Get
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
```

```
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iAMA

Fonksiyon, Adaptive Moving Average göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iAMA(
   string               symbol,              // sembol ismi
   ENUM_TIMEFRAMES      period,              // periyot
   int                  ama_period,          // AMA için ortalama periyodu
   int                  fast_ma_period,      // hızlı hareketli ortalama periyodu
   int                  slow_ma_period,      // yavaş hareketli ortalama periyodu
   int                  ama_shift,           // göstergenin yatay kaydırma değeri
   ENUM_APPLIED_PRICE   applied_price        // işleyicinin veya fiyatın tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

ama_period

[in] Etkinlik katsayısının hesaplama periyodu.

fast_ma_period

[in] Düzleştirme katsayısının hesaplanması için, hızlı ortalama periyodu.

slow_ma_period

[in] Düzleştirme katsayısının hesaplanması için, trend yokluğundaki, yavaş ortalama periyodu.

ama_shift

[in] Fiyat çizelgesine göre, göstergenin kaydırma değeri.

applied_price

[in] Kullanılan fiyat. ENUM_APPLIED_PRICE sayımındaki fiyat sabitlerinden biri veya başka bir göstergenin tanıtıcı değeri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iAMA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
```

```
//|                                                  https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla
#property description "Diğer tüm parametreler AMA'da olduğu gibidir."

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iAMA grafiği
#property indicator_label1  "iAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAMA,              // iAMA kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iAMA;          // fonksiyon tipi
input string                symbol=" ";              // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;   // zaman aralığı
input int                   ama_period=15;           // hesaplama periyodu
input int                   fast_ma_period=2;        // Hızlı ortalama periyodu
input int                   slow_ma_period=30;       // Yavaş ortalama periyodu
input int                   ama_shift=0;             // yatay kaydırma değeri
input ENUM_APPLIED_PRICE    applied_price;           // fiyat tipi
//--- gösterge tamponu
double          iAMABuffer[];
//--- iAMA göstergesinin tanıtıcı değerini depolamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Adaptive Moving Average göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
```

```
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,iAMABuffer,INDICATOR_DATA);
//--- kaydırma değerini ayarla
   PlotIndexSetInteger(0,PLOT_SHIFT,ama_shift);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iAMA)
      handle=iAMA(name,period,ama_period,fast_ma_period,slow_ma_period,ama_shift,appli
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      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;
      //--- fiyat tipi
      pars[4].type=TYPE_INT;
      pars[4].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_AMA,5,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iAMA göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturulama
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
```

```
     }
//--- Adaptive Moving Average göstergesinin hesaplandığı sembol ve zaman aralığı değe
   short_name=StringFormat("iAMA(%s/%s,%d,%d,%d,d)",name,EnumToString(period),ama_peri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iAMA göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iAMA göstergesinin değerler
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iAMABuffer dizisinin büyüklüğü iAMA gösterg
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- dizileri, Adaptive Moving Average göstergesinin değerleriyle doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iAMABuffer,ama_shift,handle,values_to_copy)) return(0);
//--- mesajı oluştur
```

```
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Adaptive Moving Average göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iAMA göstergesi ile, gösterge tamponlarının doldurulması         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // AMA çizgisinin gösterge tamponu
                         int a_shift,           // AMA çizgisinin kaydırma değeri
                         int ind_handle,        // iAMA göstergesinin tanıtıcı değeri
                         int amount             // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iAMABuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,-a_shift,amount,ama_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iAMA göstergesi kopyalanamadı, hata kodu %d",GetLastError());
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iAO

Fonksiyon, Awesome oscillator göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iAO(
   string           symbol,     // sembol ismi
   ENUM_TIMEFRAMES  period      // periyot
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iAO.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- iAO grafiği
#property indicator_label1  "iAO"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen,clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
```

```
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iAO,               // iAO kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iAO;          // fonksiyon tipi
input string                symbol=" ";             // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // zaman aralığı
//--- gösterge tamponları
double          iAOBuffer[];
double          iAOColors[];
//--- iAO göstergesinin değerlerini depolamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Awesome Oscillator göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iAOBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iAOColors,INDICATOR_COLOR_INDEX);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iAO)
      handle=iAO(name,period);
   else
      handle=IndicatorCreate(name,period,IND_AO);
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
```

```
      {
       //--- hatayı belirt ve hata kodunu al
       PrintFormat("iAO göstergesi, %s/%s sembolü için oluşturulamadı, hata kodu %d",
                   name,
                   EnumToString(period),
                   GetLastError());
       //--- gösterge erken durduruldu
       return(INIT_FAILED);
      }
//--- Awesome Oscillator göstergesinin hesaplandığı sembol ve zaman aralığı değerleri
   short_name=StringFormat("iAO(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iAO göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iAO göstergesinin değerleri
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iAOBuffer dizisinin büyüklüğü iAO gösterges
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
```

```
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iAOBuffer ve iAOColors dizilerini, Awesome Oscillator indicator göstergesinin de
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArraysFromBuffer(iAOBuffer,iAOColors,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Accelerator Oscillator göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iAO göstergesi ile, gösterge tamponlarının doldurulması          |
//+------------------------------------------------------------------+
bool FillArraysFromBuffer(double &values[],        // Awesome Oscillator değerleri içi
                          double &color_indexes[], // renk tamponu (renk indislerini d
                          int ind_handle,          // iAO göstergesinin tanıtıcı değer
                          int amount               // kopyalanan değerlerin sayısı
                          )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iAOBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iAO göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastErro
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- renk indislerini kopyala
   if(CopyBuffer(ind_handle,1,0,amount,color_indexes)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iAO göstergesinden renk değerlerinin kopyalanması başarısız oldu, h
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
```

```
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iATR

Average True Range göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iATR(
   string           symbol,        // sembol ismi
   ENUM_TIMEFRAMES  period,        // periyot
   int              ma_period      // ortalama periyodu
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

ma_period

[in] Gösterge hesabı için gereken ortalama periyodu.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iATR.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iATR grafiği
#property indicator_label1  "iATR"
#property indicator_type1   DRAW_LINE
```

```
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iATR,// iATR kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input int                   atr_period=14;          // hesaplama periyodu
input Creation              type=Call_iATR;          // fonksiyon tipi
input string                symbol=" ";             // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // zaman aralığı
//--- gösterge tamponu
double          iATRBuffer[];
//--- iAC göstergesinin tanıtıcı değerini depolamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Average True Range göstergesindeki değerlerin sayısını saklayacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iATRBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iATR)
      handle=iATR(name,period,atr_period);
   else
     {
```

```
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[1];
      pars[0].type=TYPE_INT;
      pars[0].integer_value=atr_period;
      handle=IndicatorCreate(name,period,IND_ATR,1,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("%s/%s sembolü için iATR göstergesinin tanıtıcı değeri oluşturulamad
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Average True Range göstergesinin hesaplandığı sembol ve zaman aralığı değerleri
   short_name=StringFormat("iATR(%s/%s, period=%d)",name,EnumToString(period),atr_peri
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iATR göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iATR göstergesinin değerleri
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
```

```
     {
      //--- verilen sembol ve periyot için iATRBuffer dizisinin büyüklüğü iATR gösterg
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iATRBuffer dizisini Average True Range göstergesinin değerleri ile doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iATRBuffer,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Accelerator Oscillator göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iATR göstergesi ile, gösterge tamponlarının doldurulması         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // ATR değerleri için gösterge tamponu
                         int ind_handle,    // iATR göstergesinin tanıtıcı değeri
                         int amount         // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iATRBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iATR göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastErr
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
```

```
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iBearsPower

Fonksiyon, Bears Power göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iBearsPower(
   string              symbol,           // sembol ismi
   ENUM_TIMEFRAMES     period,           // periyot
   int                 ma_period,        // ortalama periyodu
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

ma_period

[in] Gösterge hesabı için gereken ortalama periyodu.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                              Demo_iBearsPower.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                               https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iBearsPower grafiği
#property indicator_label1  "iBearsPower"
#property indicator_type1   DRAW_HISTOGRAM
```

```
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBearsPower,       // iBearsPower kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation             type=Call_iBearsPower;  // fonksiyonun tipi
input int                  ma_period=13;           // hareketli ortalama periyodu
input string               symbol=" ";             // sembol
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;  // zaman aralığı
//--- gösterge tamponu
double         iBearsPowerBuffer[];
//--- iBearsPower göstergesinin değerlerini depolamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Bears Power göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iBearsPowerBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iBearsPower)
      handle=iBearsPower(name,period,ma_period);
   else
     {
```

```
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[1];
      //--- ortalama periyodu
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BEARS,1,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iBearsPower göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşt
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Bears Power göstergesinin hesaplandığı sembol ve zaman aralığı değerlerini göste
   short_name=StringFormat("iBearsPower(%s/%s, period=%d)",name,EnumToString(period),r
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iBearsPower göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iBearsPower göstergesinin de
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
```

```
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iBearsPowerBuffer dizisinin büyüklüğü iBear
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iBearsPowerBuffer dizisini Bears Power göstergesinin değerleri ile doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iBearsPowerBuffer,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Bears Power göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iBearsPower göstergesi ile, gösterge tamponlarının doldurulması  |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // Bears Power değerleri için gösterge tar
                         int ind_handle,    // iBearsPower göstergesinin tanıtıcı değe
                         int amount         // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iBearsPowerBufferarray dizisinin bir kısmını 0 indisli tamponun değerleriyle dol
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iBearsPower göstergesinin verileri kopyalanamadı, hata kodu %d",Get
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
```

```
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iBands

Bollinger Bands® göstergesinin tanıtıcı değerine dönüş yapar.

```
int  iBands(
   string               symbol,           // sembol ismi
   ENUM_TIMEFRAMES      period,           // periyot
   int                  bands_period,     // ortalama çizgisinin hesabı için periyot
   int                  bands_shift,      // göstergenin yatay kaydırma değeri
   double               deviation,        // sapma değeri
   ENUM_APPLIED_PRICE   applied_price     // fiyat veya işleyici tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

bands_period

[in] göstergenin ana çizgisi için ortalama periyodu.

bands_shift

[in] Fiyat çizelgesine göre, göstergenin kaydırma değeri.

deviation

[in] Ana çizgiye göre sapma değeri

applied_price

[in] Kullanılan fiyat. ENUM_APPLIED_PRICE sayımındaki fiyat sabitlerinden biri veya başka bir göstergenin tanıtıcı değeri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Not

Tampon numaraları şu şekildedir: 0 - BASE_LINE, 1 - UPPER_BAND, 2 - LOWER_BAND

Örnek:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iBands.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
```

```
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_chart_window
#property indicator_buffers 3
#property indicator_plots   3
//--- Upper grafiği
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Lower grafiği
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrMediumSeaGreen
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- Middle grafiği
#property indicator_label3  "Middle"
#property indicator_type3   DRAW_LINE
#property indicator_color3  clrMediumSeaGreen
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBands,             // iBands kullan
   Call_IndicatorCreate     // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iBands;          // fonksiyon tipi
input int                   bands_period=20;           // hareketli ortalama tipi
input int                   bands_shift=0;             // kaydırma değeri
input double                deviation=2.0;             // sapma değeri
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // fiyat tipi
input string                symbol=" ";                // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // zaman aralığı
//--- gösterge tamponları
double         UpperBuffer[];
```

```
double         LowerBuffer[];
double         MiddleBuffer[];
//--- iBands göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Bollinger Bands göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
   SetIndexBuffer(2,MiddleBuffer,INDICATOR_DATA);
//--- çizgilerin kaydırma değerlerini ayarla
   PlotIndexSetInteger(0,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,bands_shift);
   PlotIndexSetInteger(2,PLOT_SHIFT,bands_shift);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iBands)
      handle=iBands(name,period,bands_period,bands_shift,deviation,applied_price);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[4];
      //--- hareketli ortalama periyodu
      pars[0].type=TYPE_INT;
      pars[0].integer_value=bands_period;
      //--- kaydırma değeri
      pars[1].type=TYPE_INT;
      pars[1].integer_value=bands_shift;
      //--- standart sapma değeri
      pars[2].type=TYPE_DOUBLE;
```

```
      pars[2].double_value=deviation;
      //--- fiyat tipi
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_BANDS,4,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iBands göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturula
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Bollinger Bands göstergesinin hesaplandığı sembol ve zaman aralığı değerlerini
   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);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iBands göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iBands göstergesinin değerle
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
```

```
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için gösterge tamponlarının büyüklüğü iBands gös
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- diziyi, Bollinger Bands göstergesinin değerleri ile doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArraysFromBuffers(MiddleBuffer,UpperBuffer,LowerBuffer,bands_shift,handle,v
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Bollinger Bands göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iBands göstergesi ile, gösterge tamponlarının doldurulması       |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &base_values[],     // Bollinger Bands orta çizgisi
                           double &upper_values[],    // üst sınır için gösterge tampo
                           double &lower_values[],    // alt sınır için gösterge tampo
                           int shift,                 // kaydırma değeri
                           int ind_handle,            // iBands göstergesinin tanıtıcı
                           int amount                 // kopyalanan değerler
                           )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- MiddleBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur index
   if(CopyBuffer(ind_handle,0,-shift,amount,base_values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iBands göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastE
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
```

```
//--- UpperBuffer dizisinin bir kısmını 1 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,1,-shift,amount,upper_values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iBands göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastE
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }

//--- LowerBuffer dizisinin bir kısmını 2 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,2,-shift,amount,lower_values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iBands göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastE
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iBullsPower

Fonksiyon, Bulls Power göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iBullsPower(
   string               symbol,            // sembol ismi
   ENUM_TIMEFRAMES      period,            // periyot
   int                  ma_period,         // ortalama periyodu
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

ma_period

[in] Gösterge hesabı için kullanılacak ortalama periyodu.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                              Demo_iBullsPower.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iBullsPower grafiği
#property indicator_label1  "iBullsPower"
#property indicator_type1   DRAW_HISTOGRAM
```

```
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBullsPower,       // iBullsPower kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iBullsPower;  // fonksiyonun tipi
input int                   ma_period=13;           // hareketli ortalama periyodu
input string                symbol=" ";             // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // zaman aralığı
//--- gösterge tamponu
double          iBullsPowerBuffer[];
//--- iBullsPower göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Bulls Power göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iBullsPowerBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iBullsPower)
      handle=iBullsPower(name,period,ma_period);
   else
     {
```

```
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[1];
      //--- hareketli ortalama periyodu
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_BULLS,1,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iBullsPower göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşt
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Bulls Power göstergesinin hesaplandığı sembol ve zaman aralığı değerlerini göste
   short_name=StringFormat("iBullsPower(%s/%s, period=%d)",name,EnumToString(period),r
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iBullsPower göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iBullsPower göstergesinin de
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
```

```
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iBullsPowerBuffer dizisinin büyüklüğü iBull
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iBullsPowerBuffer dizisini Bulls Power göstergesinin değerleri ile doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iBullsPowerBuffer,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Bulls Power göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iBullsPower göstergesi ile, gösterge tamponlarının doldurulması  |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // Bulls Power değerleri için gösterge tar
                         int ind_handle,    // iBullsPower göstergesinin tanıtıcı değe
                         int amount         // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iBullsPowerBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iBullsPower göstergesinin verileri kopyalanamadı, hata kodue %d",Ge
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
```

```
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
//+------------------------------------------------------------------+
```

iCCI

Commodity Channel Index göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iCCI(
   string               symbol,            // sembol ismi
   ENUM_TIMEFRAMES      period,            // periyot
   int                  ma_period,         // ortalama periyodu
   ENUM_APPLIED_PRICE   applied_price      // fiyatın veya işleyicinin tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

ma_period

[in] Gösterge hesabı için kullanılacak ortalama periyodu.

applied_price

[in] Kullanılan fiyat. ENUM_APPLIED_PRICE sayımındaki fiyat sabitlerinden biri veya başka bir göstergenin tanıtıcı değeri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iCCI.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_separate_window
```

```
#property indicator_buffers 1
#property indicator_plots   1
//--- iCCI grafiği
#property indicator_label1  "iCCI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- gösterge penceresindeki yatay seviyeler
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iCCI,              // iCCI kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iCCI;               // fonksiyon tipi
input int                   ma_period=14;                 // hareketli ortalama periyo
input ENUM_APPLIED_PRICE    applied_price=PRICE_TYPICAL;  // fiyat tipi
input string                symbol=" ";                   // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // zaman aralığı
//--- gösterge tamponu
double         iCCIBuffer[];
//--- iCCI göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Commodity Channel Index göstergesindeki değerlerin sayısını akılda tutacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iCCIBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
```

```
      {
       //--- göstergenin tutturulduğu çizelge sembolünü al
       name=_Symbol;
      }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iCCI)
      handle=iCCI(name,period,ma_period,applied_price);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[2];
      //--- hareketli ortalama periyodu
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- fiyat tipi
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_CCI,2,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iCCI göstergesinin tanıtıcı değeri %s/%s sembolü için oluşturulamac
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Bollinger Bands® göstergesinin hesaplandığı sembol ve zaman aralığı değerlerini
   short_name=StringFormat("iCCI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iCCI göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iCCI göstergesinin değerleri
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iCCIBuffer dizisinin büyüklüğü iCCI gösterg
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iCCIBuffer dizisini Commodity Channel Index göstergesinin değerleriyle doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iCCIBuffer,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Commodity Channel Index göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iCCI göstergesi ile, gösterge tamponlarının doldurulması         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // Commodity Channel Index değerleri için
                         int ind_handle,    // iCCI göstergesinin tanıtıcı değeri
                         int amount         // kopyalanan değerlerin sayısı
                         )
  {
```

```
//--- hata kodunu sıfırla
   ResetLastError();
//--- iCCIBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iCCI göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastEr
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iChaikin

Fonksiyon, Chaikin Oscillator göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iChaikin(
   string               symbol,              // sembol ismi
   ENUM_TIMEFRAMES      period,              // periyot
   int                  fast_ma_period,      // hızlı periyot
   int                  slow_ma_period,      // yavaş periyot
   ENUM_MA_METHOD       ma_method,           // düzleştirme tipi
   ENUM_APPLIED_VOLUME  applied_volume       // hacim tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

fast_ma_period

[in] Hesaplamada kullanılacak hızlı ortalama periyodu.

slow_ma_period

[in] Hesaplamada kullanılacak yavaş ortalama periyodu.

ma_method

[in] Düzleştirme tipi. ENUM_MA_METHOD sayımının ortalama sabitlerinden olabilir.

applied_volume

[in] Kullanılan hacim. ENUM_APPLIED_VOLUME sabitlerinden biri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                Demo_iChaikin.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iChaikin grafiği
#property indicator_label1  "iChaikin"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iChaikin,          // iChaikin kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iChaikin;           // fonksiyon tipi
input int                   fast_ma_period=3;             // hızlı hareketli ortalama p
input int                   slow_ma_period=10;            // yavaş hareketli ortalama p
input ENUM_MA_METHOD        ma_method=MODE_EMA;           // düzleştirme tipi
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // hacim tipi
input string                symbol=" ";                   // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // zaman aralığı
//--- gösterge tamponu
double         iChaikinBuffer[];
//--- iChaikin göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Chaikin Oscillator göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iChaikinBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
```

```
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iChaikin)
      handle=iChaikin(name,period,fast_ma_period,slow_ma_period,ma_method,applied_volu
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[4];
      //--- hızlı HO periyodu
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ma_period;
      //--- yavaş HO periyodu
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ma_period;
      //--- düzleştirme tipi
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- hacim tipi
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_CHAIKIN,4,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iChaikin göstergesinin tanıtıcı değeri, %s/%s sembolü için oluştur
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Chaikin Oscillator göstergesinin hesaplandığı sembol ve zaman aralığı değerleri
   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);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
```

```
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iChaikin göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iChaikin göstergesinin değe
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iChaikinBuffer dizisinin büyüklüğü iChaikin
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iChaikinBuffer dizisini Chaikin Oscillator göstergesinin değerleri ile doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iChaikinBuffer,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
```

```
//--- Chaikin Oscillator göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iChaikin göstergesi ile, gösterge tamponlarının doldurulması     |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // Chaikin Oscillator değerleri için göste
                         int ind_handle,    // iChaikin göstergesinin tanıtıcı değeri
                         int amount         // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iChaikinBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iChaikin göstergesinden veriler kopyalanamadı, hata kodu %d",GetLas
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iCustom

Belirtilen özel göstergenin tanıtıcı değerine dönüş yapar.

```
int  iCustom(
   string           symbol,     // sembol ismi
   ENUM_TIMEFRAMES  period,     // periyot
   string           name        // klasörün veya özel göstergenin ismi
   ...                          // gösterge giriş parametrelerinin listesi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

name

[in] Özel gösterge adı. Eğer ad ters eğik çizgi '\' ile başlıyorsa, EX5 gösterge dosyası MQL5\Indicators gösterge kök dizininde aranır. Bu nedenle, iCustom(Symbol(), Period(), "\FirstIndicator"...) çağrıldığında, gösterge MQL5\Indicators\FirstIndicator.ex5 olarak indirilir. Eğer yolda dosya yoksa, 4802 hatası (ERR_INDICATOR_CANNOT_CREATE) oluşur.

Yol '\' ile başlamıyorsa, gösterge aşağıdaki gibi aranır ve indirilir:

- İlk olarak, çağrı programının EX5 dosyasının bulunduğu klasörde EX5 gösterge dosyası aranır. Örneğin, CrossMA.EX5 uzman danışmanı MQL5\Experts\MyExperts içerisinde bulunmaktadır ve iCustom çağrısını (Symbol(), Period(), "SecondIndicator"...) içermektedir. Bu durumda, gösterge MQL5\Experts\MyExperts\SecondIndicator.ex5 olarak aranır.
- Gösterge aynı dizinde bulunamazsa, arama MQL5\Indicators gösterge kök dizininde gerçekleştirilir. Başka bir deyişle, MQL5\Indicators\SecondIndicator.ex5 dosyası aranır. Gösterge hala bulunamazsa, fonksiyon INVALID_HANDLE geri döndürür ve 4802 hatası (ERR_INDICATOR_CANNOT_CREATE) tetiklenir.

Göstergenin yolu bir alt dizinde ayarlanmışsa (örneğin, MyIndicators\ThirdIndicator), arama ilk olarak çağrı programı klasöründe (uzman danışman MQL5\Experts\MyExperts klasöründe bulunur) MQL5\Experts\MyExperts\MyIndicators\ThirdIndicator.ex5 dosyası için gerçekleştirilir. Başarısız olursa, MQL5\Indicators\MyIndicators\ThirdIndicator.ex5 dosyası aranır. Yolda ayırıcı olarak çift ters eğik çizgiyi '\\' kullandığınızdan emin olun, örneğin iCustom(Symbol(), Period(), "MyIndicators\\ThirdIndicator"...)

...

[in] Virgüle ayrılmış şekilde, özel göstergenin giriş-parametreleri. parametrelerin tipleri ve sıraları uyumlu olmalıdır. Eğer belirtilen bir parametre değeri yoksa, varsayılan değerler kullanılır.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden

temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

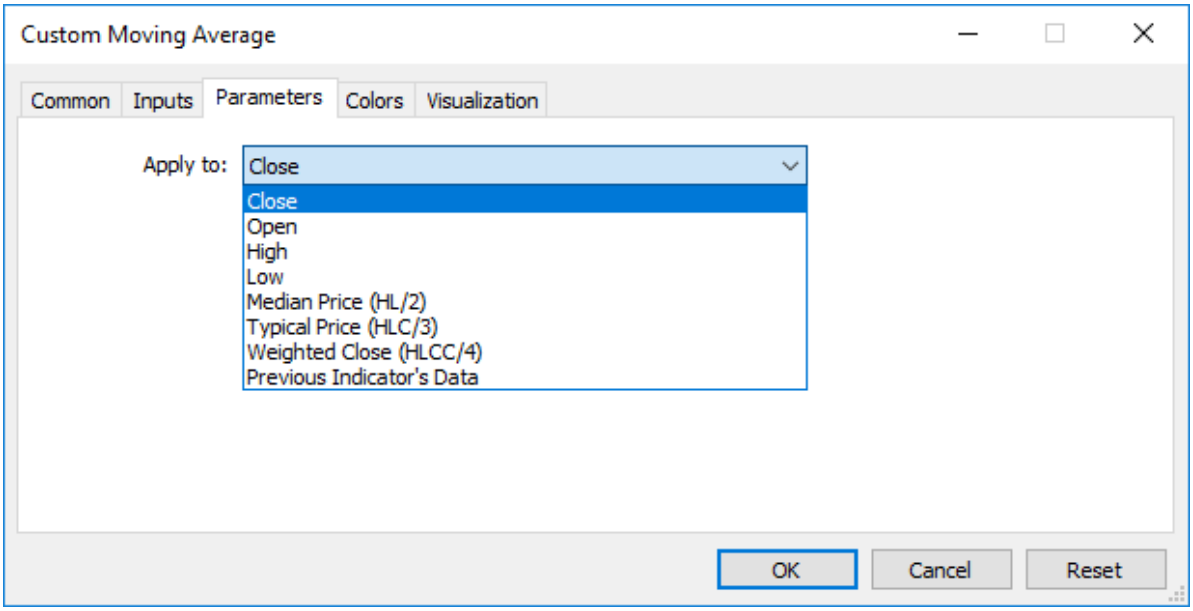
Not

Bir özel gösterge, (EX5 uzantısı ile) derlenmeli ve müşteri terminalinin MQL5/Indicators klasörüne - veya bir alt klasöre - yerleştirilmelidir.

Sınama gerektiren göstergeler, eğer karşılık gelen parametre bir sabit dizgi ile ayarlanmışsa, iCustom() fonksiyonunun çağrısıyla otomatik olarak tanımlanırlar. Tüm diğer durumlarda (IndicatorCreate() fonksiyonunun kullanımı veya sabit olmayan bir dizginin gösterge ismini belirleyen bir parametrede kullanımı) #property tester_indicator özelliği gereklidir:

```
#property tester_indicator "gösterge_ismi.ex5"
```

Eğer göstergede ilk çağrı biçimi kullanılmışsa, o zaman gösterge başlangıcında "Parameters" sekmesinde, hesaplama için ek veriler belirtebilirsiniz. "Apply to" parametresi açıkça seçilmemişse, varsayılan hesaplama "Close" (kapanış) fiyatlarıyla yapılır.



Bir MQL5 programının içinden bir özel gösterge çağrıldığında, özel göstergenin girdi değişkenlerinden sonra, Applied_Price parametresi veya başka bir göstergenin tanıtıcı değeri son olarak geçirilmelidir.

Ayrıca Bakınız

Program Özellikleri, Zaman Serileri ve Göstergelere Erişim, IndicatorCreate(), IndicatorRelease()

Örnek:

```
#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
//--- giriş parametreleri
input int MA_Period=21;
input int MA_Shift=0;
input ENUM_MA_METHOD MA_Method=MODE_SMA;
//--- gösterge tamponları
double         Label1Buffer[];
//--- Custom Moving Average.mq5 özel göstergesinin tanıtıcı değeri
int MA_handle;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- gösterge tamponlarının eşlenmesi
   SetIndexBuffer(0,Label1Buffer,INDICATOR_DATA);
   ResetLastError();
   MA_handle=iCustom(NULL,0,"Examples\\Custom Moving Average",
                     MA_Period,
                     MA_Shift,
                     MA_Method,
                     PRICE_CLOSE // kapanış fiyatlarını kullanarak
                     );
   Print("MA_handle = ",MA_handle,"  hata = ",GetLastError());
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- Custom Moving Average göstergesinin değerlerini gösterge tamponuna kopyala
   int copy=CopyBuffer(MA_handle,0,0,rates_total,Label1Buffer);
   Print("copy = ",copy,"    rates_total = ",rates_total);
//--- Deneme başarısız olduysa - Bunu rapor et
   if(copy<=0)
      Print("Custom Moving Average değerlerinin kopyalanması başarısız oldu");
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
```

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

iDEMA

Fonksiyon, Double Exponential Moving Average göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iDEMA(
   string               symbol,            // sembol ismi
   ENUM_TIMEFRAMES      period,            // periyot
   int                  ma_period,         // ortalama periyodu
   int                  ma_shift,          // yatay kaydırma değeri
   ENUM_APPLIED_PRICE   applied_price      // fiyat veya işleyici tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

ma_period

[in] Hesaplamalar için gereken ortalama periyodu (çubuk sayısı olarak).

ma_shift

[in] Fiyat çizelgesine göre, göstergenin kaydırma değeri.

applied_price

[in] Kullanılan fiyat. ENUM_APPLIED_PRICE sayımındaki fiyat sabitlerinden biri veya başka bir göstergenin tanıtıcı değeri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iDEMA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
```

```
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iDEMA grafiği
#property indicator_label1  "iDEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iDEMA,              // iDEMA kullan
   Call_IndicatorCreate     // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iDEMA;           // fonksiyonun tipi
input int                   ma_period=14;              // hareketli ortalama periyodu
input int                   ma_shift=0;                // kaydırma değeri
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // fiyat tipi
input string                symbol=" ";                // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // zaman aralığı
//--- gösterge tamponu
double          iDEMABuffer[];
//--- iDEMA göstergesinin tanıtıcı değerini saklamak için bir değişken
int     handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Double Exponential Moving Average göstergesindeki değerlerin sayısını koruyacağ
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iDEMABuffer,INDICATOR_DATA);
   //--- kaydırma değerini ayarla
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
```

```
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iDEMA)
      handle=iDEMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[3];
      //--- hareketli ortalama periyodu
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- kaydırma değeri
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- fiyat tipi
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_DEMA,3,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iDEMA göstergesinin tanıtıcı değeri %s/%s sembolü için oluşturulama
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Double Exponential Moving Average göstergesinin hesaplandığı sembol ve zaman ara
   short_name=StringFormat("iDEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iDEMA göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iDEMA göstergesinin değerler
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iDEMABuffer dizisinin büyüklüğü iDEMA göste
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iDEMABuffer dizisini, Double Exponential Moving Average göstergesinin değerleriy
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iDEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Double Exponential Moving Average göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
```

```
//+------------------------------------------------------------------+
//| iDEMA göstergesi ile, gösterge tamponlarının doldurulması        |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // Double Exponential Moving Average değe
                         int shift,         // kaydırma değeri
                         int ind_handle,    // iDEMA göstergesinin tanıtıcı değeri
                         int amount         // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iDEMABuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iDEMA göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastE
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iDeMarker

Fonksiyon, DeMarker göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iDeMarker(
   string           symbol,        // sembol ismi
   ENUM_TIMEFRAMES  period,        // periyot
   int              ma_period      // ortalama periyodu
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

ma_period

[in] Hesaplamalar için gereken ortalama periyodu (çubuk sayısı olarak).

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                Demo_iDeMarker.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iDeMarker grafiği
#property indicator_label1  "iDeMarker"
#property indicator_type1   DRAW_LINE
```

```
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- gösterge penceresindeki yatay seviyeler
#property indicator_level1  0.3
#property indicator_level2  0.7
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iDeMarker,          // iDeMarker kullan
   Call_IndicatorCreate     // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iDeMarker;       // fonksiyon tipi
input int                   ma_period=14;              // hareketli ortalama periyodu
input string                symbol=" ";                // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // zaman aralığı
//--- gösterge tamponu
double          iDeMarkerBuffer[];
//--- iDeMarker göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- DeMarker göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iDeMarkerBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iDeMarker)
```

```
      handle=iDeMarker(name,period,ma_period);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[1];
      //--- hareketli ortalama periyodu
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_DEMARKER,1,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iDeMarker göstergesinin tanıtıcı değeri %s/%s sembolü için oluştur
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- DeMarker göstergesinin hesaplandığı sembol ve zaman aralığı değerlerini göster
   short_name=StringFormat("iDeMarker(%s/%s, period=%d)",name,EnumToString(period),ma_
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iDeMarker göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
```

```
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iDeMarker göstergesinin değe
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iDeMarkerBuffer dizisinin büyüklüğü iDeMark
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iDeMarkerBuffer dizisini DeMarker göstergesinin değerleri ile doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iDeMarkerBuffer,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- DeMarker göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iDeMarker göstergesi ile, gösterge tamponlarının doldurulması    |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // DeMarker değerleri için gösterge tampon
                         int ind_handle,    // iDeMarker göstergesinin tanıtıcı değeri
                         int amount         // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iDeMarkerBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iDeMarker göstergesinin verileri kopyalanamadı, hata kodu %d",GetLa
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
```

```
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iEnvelopes

Fonksiyon, Envelopes göstergesinin tanıtıcı değerine dönüş yapar.

```
int  iEnvelopes(
   string               symbol,           // sembol ismi
   ENUM_TIMEFRAMES      period,           // periyot
   int                  ma_period,        // ortalama çizgisinin hesabı için periyot
   int                  ma_shift,         // göstergenin yatay kaydırma değeri
   ENUM_MA_METHOD       ma_method,        // düzleştirme tipi
   ENUM_APPLIED_PRICE   applied_price,    // fiyat veya işleyici tipi
   double               deviation         // sınırların orta çizgiden sapması (yüzde
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

ma_period

[in] Ana çizgi için ortalama periyodu.

ma_shift

[in] Fiyat çizelgesine göre, göstergenin kaydırma değeri.

ma_method

[in] Düzleştirme tipi. ENUM_MA_METHOD değerlerinden biri olabilir.

applied_price

[in] Kullanılan fiyat. ENUM_APPLIED_PRICE sayımındaki fiyat sabitlerinden biri veya başka bir göstergenin tanıtıcı değeri olabilir.

deviation

[in] Ana çizgiden sapma (yüzde olarak).

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Not

Tampon numaraları: 0 - UPPER_LINE, 1 - LOWER_LINE.

Örnek:

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

```
//|                                                  Demo_iEnvelopes.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                  https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
//--- Upper grafiği
#property indicator_label1  "Upper"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Lower grafiği
#property indicator_label2  "Lower"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iEnvelopes,         // iEnvelopes kullan
   Call_IndicatorCreate     // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iEnvelopes;      // fonksiyon tipi
input int                   ma_period=14;              // hareketli ortalama periyodu
input int                   ma_shift=0;                // kaydırma değeri
input ENUM_MA_METHOD        ma_method=MODE_SMA;        // düzeltme tipi
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // fiyat tipi
input double                deviation=0.1;             // sınırların hareketli ortalama
input string                symbol=" ";                // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // zaman aralığı
//--- gösterge tamponu
double         UpperBuffer[];
double         LowerBuffer[];
//--- iEnvelopes göstergesinin tanıtıcı değerini saklamak için bir değişken
```

```
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Envelopes göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,UpperBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,LowerBuffer,INDICATOR_DATA);
//--- çizgilerin kaydırma değerlerini ayarla
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
   PlotIndexSetInteger(1,PLOT_SHIFT,ma_shift);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iEnvelopes)
      handle=iEnvelopes(name,period,ma_period,ma_shift,ma_method,applied_price,deviati
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[5];
      //--- hareketli ortalama periyodu
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- kaydırma değeri
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- düzleştirme tipi
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- fiyat tipi
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      //--- fiyat tipi
```

```
      pars[4].type=TYPE_DOUBLE;
      pars[4].double_value=deviation;
      handle=IndicatorCreate(name,period,IND_ENVELOPES,5,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iEnvelopes göstergesinin tanıtıcı değeri, %s/%s sembolü için oluştu
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Envelopes göstergesinin hesaplandığı sembol ve zaman aralığı değerlerini göster
   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);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iEnvelopes göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iEnvelopes göstergesinin değ
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
```

```
      //--- verilen sembol ve periyot için UpperBuffer dizisinin büyüklüğü iEnvelopes
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- UpperBuffer ve LowerBuffer dizilerini, Envelopes göstergesinin değerleriyle dold
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArraysFromBuffers(UpperBuffer,LowerBuffer,ma_shift,handle,values_to_copy))
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Envelopes göstergesinin değerlerinin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iEnvelopes göstergesi ile, gösterge tamponlarının doldurulması   |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &upper_values[],     // üst sınırın gösterge tamponu
                           double &lower_values[],     // alt sınırın gösterge tamponu
                           int shift,                  // kaydırma değeri
                           int ind_handle,             // iEnvelopes göstergesinin tan
                           int amount                  // kopyalanan değerler
                           )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- UpperBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,-shift,amount,upper_values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iEnvelopes göstergesinin verileri kopyalanamadı, hata kodu %d",GetL
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- LowerBuffer dizisinin bir kısmını 1 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,1,-shift,amount,lower_values)<0)
     {
```

```
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iEnvelopes göstergesinin verileri kopyalanamadı, hata kodu %d",GetL
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iForce

Fonksiyon, Force Index göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iForce(
   string               symbol,            // sembol ismi
   ENUM_TIMEFRAMES      period,            // periyot
   int                  ma_period,         // ortalama periyodu
   ENUM_MA_METHOD       ma_method,         // düzleştirme tipi
   ENUM_APPLIED_VOLUME  applied_volume     // kullanılacak hacim tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

ma_period

[in] Gösterge hesabı için kullanılacak ortalama periyodu.

ma_method

[in] Düzleştirme tipi. ENUM_MA_METHOD değerlerinden biri olabilir.

applied_volume

[in] Kullanılan hacim. ENUM_APPLIED_VOLUME sayımının değerlerinden biri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iForce.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
```

```
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iForce çizimi
#property indicator_label1  "iForce"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iForce,            // iForce kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iForce;             // fonksiyon tipi
input int                   ma_period=13;                 // ortalama periyodu
input ENUM_MA_METHOD        ma_method=MODE_SMA;           // düzleştirme tipi
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // hacim tipi
input string                symbol=" ";                   // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // zaman aralığı
//--- gösterge tamponu
double         iForceBuffer[];
//--- iForce göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Force göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iForceBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
```

```
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iForce)
      handle=iForce(name,period,ma_period,ma_method,applied_volume);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[3];
      //--- hareketli ortalama periyodu
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- düzleştirme tipi
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_method;
      //--- hacim tipi
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_volume;
      //--- fiyat tipi
      handle=IndicatorCreate(name,period,IND_FORCE,3,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iForce göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturula
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Force göstergesinin hesaplandığı sembol ve zaman aralığı değerlerini göster
   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);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iForce göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iForce göstergesinin değerle
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iForceBuffer dizisinin büyüklüğü iForce gös
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iForceBuffer dizisini Force göstergesinin değerleri ile doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iForceBuffer,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Force göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iForce göstergesi ile, gösterge tamponlarının doldurulması       |
//+------------------------------------------------------------------+
```

```
bool FillArrayFromBuffer(double &values[],  // Force Index değerleri için gösterge tar
                         int ind_handle,    // iForce göstergesinin tanıtıcı değeri
                         int amount         // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iForceBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iForce göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastE
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iFractals

Fonksiyon, Fractals göstergesinin tanıtıcı değerine dönüş yapar.

```
int  iFractals(
   string           symbol,     // sembol ismi
   ENUM_TIMEFRAMES  period      // periyot
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Not

Tampon numaraları şu şekildedir: 0 - UPPER_LINE, 1 - LOWER_LINE.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                Demo_iFractals.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
#property indicator_chart_window
#property indicator_buffers 2
#property indicator_plots   2
```

```
//--- FractalUp grafiği
#property indicator_label1  "FractalUp"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrBlue
//--- FractalDown grafiği
#property indicator_label2  "FractalDown"
#property indicator_type2   DRAW_ARROW
#property indicator_color2  clrRed
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iFractals,         // iFractals kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation             type=Call_iFractals;          // fonksiyon tipi
input string               symbol=" ";                   // sembol
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // zaman aralığı
//--- gösterge tamponları
double         FractalUpBuffer[];
double         FractalDownBuffer[];
//--- iFractals göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Fractals göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,FractalUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,FractalDownBuffer,INDICATOR_DATA);
//--- Wingdings karakter kümesindeki sembolleri kullanarak PLOT_ARROW özelliği için ko
   PlotIndexSetInteger(0,PLOT_ARROW,217); // yukarı ok
   PlotIndexSetInteger(1,PLOT_ARROW,218); // aşağı ok
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
```

```
      {
       //--- göstergenin tutturulduğu çizelge sembolünü al
       name=_Symbol;
      }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iFractals)
      handle=iFractals(name,period);
   else
      handle=IndicatorCreate(name,period,IND_FRACTALS);
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iFractals göstergesinin tanıtıcı değeri, %s/%s sembolü için oluştu
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Fractals göstergesinin hesaplandığı sembol ve zaman aralığı değerlerini göster
   short_name=StringFormat("iFractals(%s/%s)",name,EnumToString(period));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iFractals göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
```

```
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iFractals göstergesinin değe
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için FractalUpBuffer dizisinin büyüklüğü iFracta
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- FractalUpBuffer ve FractalDownBuffer dizilerini Fractals göstergesinin değerleri
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArraysFromBuffers(FractalUpBuffer,FractalDownBuffer,handle,values_to_copy))
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Fractals göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iDeMarker göstergesi ile, gösterge tamponlarının doldurulması    |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &up_arrows[],        // yukarı oklar için gösterge t
                           double &down_arrows[],      // aşağı oklar için gösterge ta
                           int ind_handle,             // iFractals göstergesinin tan
                           int amount                  // kopyalanan verilerin sayısı
                           )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- FractalUpBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,up_arrows)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iFractals göstergesinin verileri FractalUpBuffer dizisine kopyalana
                  GetLastError());
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
```

```
      }
//--- FractalDownBuffer dizisinin bir kısmını 1 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,1,0,amount,down_arrows)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iFractals göstergesinin verileri FractalDownBuffer dizisine kopyala
                  GetLastError());
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iFrAMA

Fonksiyon, Fractal Adaptive Moving Average göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iFrAMA(
   string               symbol,            // sembol ismi
   ENUM_TIMEFRAMES      period,            // periyot
   int                  ma_period,         // ortalama periyodu
   int                  ma_shift,          // çizelge üzerindeki yatay kaydırma değeri
   ENUM_APPLIED_PRICE   applied_price      // fiyat veya işleyici tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

ma_period

[in] Gösterge hesabı için periyot (çubuk sayısıyla).

ma_shift

[in] Göstergenin çizelgede kaydırılması için kullanılacak değer.

applied_price

[in] Kullanılan fiyat. ENUM_APPLIED_PRICE sayımındaki fiyat sabitlerinden biri veya başka bir göstergenin tanıtıcı değeri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iFrAMA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
```

```
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iFrAMA çizimi
#property indicator_label1  "iFrAMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iFrAMA,            // iFrAMA kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iFrAMA;             // fonksiyon tipi
input int                   ma_period=14;                 // ortalama periyodu
input int                   ma_shift=0;                   // kaydırma
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE;    // fiyat tipi
input string                symbol=" ";                   // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // zaman aralığı
//--- gösterge tamponu
double         iFrAMABuffer[];
//--- iFrAMA göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Fractal Adaptive Moving Average göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iFrAMABuffer,INDICATOR_DATA);
//--- kaydırma değerini ayarla
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
```

```
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iFrAMA)
      handle=iFrAMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[3];
      //--- hareketli ortalama periyodu
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- kaydırma değeri
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- fiyat tipi
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      //--- fiyat tipi
      handle=IndicatorCreate(name,period,IND_FRAMA,3,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iFrAMA göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturula
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- iFrAMA göstergesinin hesaplandığı sembol ve zaman aralığı değerlerini göster
   short_name=StringFormat("iFrAMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
```

```
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[])
  {
//--- iFrAMA göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iFrAMA göstergesinin değerle
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iFrAMABuffer dizisinin büyüklüğü iFrAMA gös
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iFrAMABuffer dizisini Fractal Adaptive Moving Average göstergesinin değerleriyle
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iFrAMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Fractal Adaptive Moving Average göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
```

```
  }
//+------------------------------------------------------------------+
//| iFrAMA göstergesi ile, gösterge tamponlarının doldurulması       |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // Fractal Adaptive Moving Average değerle
                         int shift,         // kaydırma değeri
                         int ind_handle,    // iFrAMA göstergesinin tanıtıcı değeri
                         int amount         // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iFrAMABuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iFrAMA göstergesinin verileri kopyalanamadı, hata kodu %d %d",GetLa
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iGator

Gator göstergesinin tanıtıcı değerine dönüş yapar. Bu osilatör, Alligator (timsah) göstergesinin mavi ve kırmızı çizgileri arasındaki farkı (üst histogram) ve kırmızı ve yeşil çizgileri arasındaki farkı (alt histogram) gösterir.

```
int  iGator(
   string               symbol,            // sembol ismi
   ENUM_TIMEFRAMES      period,            // periyot
   int                  jaw_period,        // çenelerin hesabı için periyot değeri
   int                  jaw_shift,         // çeneler için yatay kaydırma değeri
   int                  teeth_period,      // dişlerin hesaplanması için periyot
   int                  teeth_shift,       // dişler için yatay kaydırma değeri
   int                  lips_period,       // dudakların hesaplanması için periyot
   int                  lips_period,       // dudakların hesaplanması için periyot
   ENUM_MA_METHOD       ma_method,         // düzleştirme tipi
   ENUM_APPLIED_PRICE   applied_price      // fiyat veya işleyici tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

jaw_period

[in] Mavi çizgi için ortalama periyodu (Timsahın çenesi).

jaw_shift

[in] fiyat çizelgesine göre mavi çizginin kaydırma değeri. Bu değer, göstergenin görsel kaydırma değeri ile doğrudan bağlantılı değildir.

teeth_period

[in] Kırmızı çizgi için ortalama periyodu (Timsahın dişleri).

teeth_shift

[in] fiyat çizelgesine göre kırmızı çizginin kaydırma değeri. Bu değer, göstergenin görsel kaydırma değeri ile doğrudan bağlantılı değildir.

lips_period

[in] Yeşil çizgi için ortalama periyodu (Timsahın dudakları).

lips_shift

[in] Fiyat çizelgesine göre yeşil çizginin kaydırma değeri. Bu değer, göstergenin görsel kaydırma değeri ile doğrudan bağlantılı değildir.

ma_method

[in] Düzleştirme tipi. ENUM_MA_METHOD değerlerinden biri olabilir.

applied_price

[in] Kullanılan fiyat. ENUM_APPLIED_PRICE sayımındaki fiyat sabitlerinden biri veya başka bir göstergenin tanıtıcı değeri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Not

Tampon numaraları: 0 - UPPER_HISTOGRAM, 1 - üst histogram için renk tamponu, 2 - LOWER_HISTOGRAM, 3 - alt histogram için renk tamponu.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iGator.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla
#property description "diğer tüm parametreler, standart Gator Oscilator'da olduğu gibi

#property indicator_separate_window
#property indicator_buffers 4
#property indicator_plots   2
//--- GatorUp çizimi
#property indicator_label1  "GatorUp"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- GatorDown çizimi
#property indicator_label2  "GatorDown"
#property indicator_type2   DRAW_COLOR_HISTOGRAM
#property indicator_color2  clrGreen, clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
```

```
enum Creation
  {
   Call_iGator,            // iGator kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iGator;       // fonksiyon tipi
input string                symbol=" ";             // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;  // zaman aralığı
input int                   jaw_period=13;          // Jaw (çene) çizgisinin periyodu
input int                   jaw_shift=8;            // Jaw çizgisinin kaydırma değeri
input int                   teeth_period=8;         // Teeth (diş) çizgisinin periyodu
input int                   teeth_shift=5;          // Teeth çizgisinin kaydırma değeri
input int                   lips_period=5;          // Lips (dudak) çizgisinin periyodu
input int                   lips_shift=3;           // Lips çizgisinin kaydırma değeri
input ENUM_MA_METHOD        MA_method=MODE_SMMA;    // Alligator çizgilerinin ortalama
input ENUM_APPLIED_PRICE    applied_price=PRICE_MEDIAN;// Alligator hesaplamasında kul
//--- gösterge tamponları
double          GatorUpBuffer[];
double          GatorUpColors[];
double          GatorDownBuffer[];
double          GatorDownColors[];
//--- iGator göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- üst ve alt histogramlar için kaydırma değeri
int shift;
//--- Gator Oscillator göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,GatorUpBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,GatorUpColors,INDICATOR_COLOR_INDEX);
   SetIndexBuffer(2,GatorDownBuffer,INDICATOR_DATA);
   SetIndexBuffer(3,GatorDownColors,INDICATOR_COLOR_INDEX);
/*
  Tüm bu kaydırma değerleri, Gator Oscilator göstergesi içinde, Alligator göstergesini
  Bu yüzden Gator göstergesini değil, Alligator çizgilerini kaydırırlar,
  sonra buradan elde edilen değerler Gator Oscillator göstergesinin hesaplanmasında ku
*/
//--- Şimdi, Jaw çizgisi ile Teeth çizgisi arasındaki farka eşit olan alt histogramın
   shift=MathMin(jaw_shift,teeth_shift);
```

```
   PlotIndexSetInteger(0,PLOT_SHIFT,shift);
//--- iki histogram bulunmasına rağmen, aynı kaydırma değeri kullanılır - bu, iGator g
   PlotIndexSetInteger(1,PLOT_SHIFT,shift);

//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   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
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[8];
      //--- Alligator çizgilerinin periyodları ve kaydırma değerleri
      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;
      //--- düzleştirme tipi
      pars[6].type=TYPE_INT;
      pars[6].integer_value=MA_method;
      //--- fiyat tipi
      pars[7].type=TYPE_INT;
      pars[7].integer_value=applied_price;
      //--- işleyiciyi oluştur
      handle=IndicatorCreate(name,period,IND_GATOR,8,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
```

```
      PrintFormat("iGator göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturula
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Gator Oscillator göstergesinin hesaplandığı sembol ve zaman aralığı değerlerini
   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);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iGator göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iGator göstergesinin değerle
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için GatorUpBuffer dizisinin büyüklüğü iGator gö
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
```

```
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- dizileri Gator Oscillator göstergesinin değerleriyle doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArraysFromBuffers(GatorUpBuffer,GatorUpColors,GatorDownBuffer,GatorDownColo
      shift,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Gator Oscillator göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iGator göstergesi ile, gösterge tamponlarının doldurulması       |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &ups_buffer[],         // üst histogram için gösterg
                           double &up_color_buffer[],    // üst histogramın fiyat indi
                           double &downs_buffer[],       // alt histogram için gösterg
                           double &downs_color_buffer[], // alt histogramın fiyat indi
                           int u_shift,                  // alt ve üst histogramlar iç
                           int ind_handle,               // iGator göstergesinin tanıt
                           int amount                    // kopyalanmış değerlerin say
                           )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- GatorUpBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,-u_shift,amount,ups_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iGator göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastE
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }

//--- GatorUpColors dizisinin bir kısmını 1 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,1,-u_shift,amount,up_color_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iGator göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastE
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
```

```
      }

//--- GatorDownBuffer dizisinin bir kısmını 2 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,2,-u_shift,amount,downs_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iGator göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastE
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }

//--- GatorDownColors dizisinin bir kısmını 3 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,3,-u_shift,amount,downs_color_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iGator göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastE
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iIchimoku

Ichimoku Kinko Hyo göstergesinin tanıtıcı değerine dönüş yapar.

```
int  iIchimoku(
   string           symbol,           // sembol ismi
   ENUM_TIMEFRAMES  period,           // periyot
   int              tenkan_sen,       // Tenkan-sen periyodu
   int              kijun_sen,        // Kijun-sen periyodu
   int              senkou_span_b     // Senkou Span B periyodu
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

tenkan_sen

[in] Tenkan Sen için ortalama periyodu.

kijun_sen

[in] Kijun Sen için ortalama periyodu.

senkou_span_b

[in] Senkou Span B için ortalama periyodu.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Not

Tampon numaraları: 0 - TENKANSEN_LINE, 1 - KIJUNSEN_LINE, 2 - SENKOUSPANA_LINE, 3 - SENKOUSPANB_LINE, 4 - CHIKOUSPAN_LINE.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                Demo_iIchimoku.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla
#property description "Diğer tüm parametreler standart Ichimoku Kinko Hyo göstergesinc

#property indicator_chart_window
#property indicator_buffers 5
#property indicator_plots   4
//--- Tenkan_sen grafiği
#property indicator_label1  "Tenkan_sen"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Kijun_sen grafiği
#property indicator_label2  "Kijun_sen"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrBlue
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- Senkou_Span grafiği
#property indicator_label3  "Senkou Span A;Senkou Span B" // bu iki alan Veri Penceres
#property indicator_type3   DRAW_FILLING
#property indicator_color3  clrSandyBrown, clrThistle
#property indicator_style3  STYLE_SOLID
#property indicator_width3  1
//--- Chikou_Span grafiği
#property indicator_label4  "Chikou_Span"
#property indicator_type4   DRAW_LINE
#property indicator_color4  clrLime
#property indicator_style4  STYLE_SOLID
#property indicator_width4  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iIchimoku,          // iIchimoku kullan
   Call_IndicatorCreate     // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iIchimoku;       // fonksiyon tipi
input int                   tenkan_sen=9;              // Tenkan-sen periyodu
input int                   kijun_sen=26;              // Kijun-sen periyodu
input int                   senkou_span_b=52;          // Senkou Span B periyodu
input string                symbol=" ";                // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // zaman aralığı
```

```
//--- gösterge tamponu
double          Tenkan_sen_Buffer[];
double          Kijun_sen_Buffer[];
double          Senkou_Span_A_Buffer[];
double          Senkou_Span_B_Buffer[];
double          Chinkou_Span_Buffer[];
//--- iIchimoku göstergesinin tanıtıcı değerini saklamak için bir değişken
int     handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Ichimoku Kinko Hyo göstergesindeki değerlerin sayısını koruyacağız
int     bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   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);
//--- kijun_sen çubuklarının Senkou Span kanalının ileriye doğru kaydırma değeri
   PlotIndexSetInteger(2,PLOT_SHIFT,kijun_sen);
//--- Chikou data Span, bir kaydırma değeri ile zaten iIchimoku içinde kayıtlı olduğu
//--- Chinkou Span çizgisine bir kaydırma değeri ayarlanması gerekmiyor
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iIchimoku)
      handle=iIchimoku(name,period,tenkan_sen,kijun_sen,senkou_span_b);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[3];
      //--- Alligator çizgilerinin periyodları ve kaydırma değerleri
      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;
      //--- işleyiciyi oluştur
      handle=IndicatorCreate(name,period,IND_ICHIMOKU,3,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iIchimoku göstergesinin tanıtıcı değeri, %s/%s sembolü için oluştu
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Ichimoku Kinko Hyo göstergesinin hesaplandığı sembol ve zaman aralığı değerleri
   short_name=StringFormat("iIchimoku(%s/%s, %d, %d ,%d)",name,EnumToString(period),
                           tenkan_sen,kijun_sen,senkou_span_b);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iIchimoku göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
```

```
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iIchimoku göstergesinin değe
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için Tenkan_sen_Buffer dizisinin büyüklüğü iIchi
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- dizileri Ichimoku Kinko Hyo indicator göstergesinin değerleriyle doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabili
   if(!FillArraysFromBuffers(Tenkan_sen_Buffer,Kijun_sen_Buffer,Senkou_Span_A_Buffer,S
      kijun_sen,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Ichimoku Kinko Hyo göstergesindeki değerlerin sayısını koruyacağız
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iIchimoku göstergesi ile, gösterge tamponlarının doldurulması    |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &tenkan_sen_buffer[],     // Tenkan-sen çizgisi için
                           double &kijun_sen_buffer[],      // Kijun_sen çizgisi için
                           double &senkou_span_A_buffer[],  // Senkou Span A çizgisi
                           double &senkou_span_B_buffer[],  // Senkou Span B çizgisi
                           double &chinkou_span_buffer[],   // Chinkou Span çizgisi iç
                           int senkou_span_shift,           // Senkou Span çizgileri
                           int ind_handle,                  // iIchimoku göstergesinin
                           int amount                       // kopyalanan değerlerin s
                           )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- Tenkan_sen_Buffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,tenkan_sen_buffer)<0)
     {
```

```
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("1. iIchimoku göstergesinin verileri kopyalanamadı, hata kodu %d",Ge
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }

//--- Kijun_sen_Buffer dizisinin bir kısmını 1 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,1,0,amount,kijun_sen_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("2. iIchimoku göstergesinin verileri kopyalanamadı, hata kodu %d",Ge
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }

//--- Chinkou_Span_Buffer dizisinin bir kısmını 2 indisli tamponun değerleriyle doldur
//--- senkou_span_shift>0 ise, tçizgi ileriye doğru senkou_span_shift çubuk kadar kayd
   if(CopyBuffer(ind_handle,2,-senkou_span_shift,amount,senkou_span_A_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("3. iIchimoku göstergesinin verileri kopyalanamadı, hata kodu %d",Ge
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }

//--- Senkou_Span_A_Buffer dizisinin bir kısmını 3 indisli tamponun değerleriyle doldu
//--- senkou_span_shift>0 ise, tçizgi ileriye doğru senkou_span_shift çubuk kadar kayd
   if(CopyBuffer(ind_handle,3,-senkou_span_shift,amount,senkou_span_B_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("4. iIchimoku göstergesinin verileri kopyalanamadı, hata kodu %d",Ge
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }

//--- Senkou_Span_B_Buffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldu
//--- Chinkou Span değerlerini kopyalarken, Chinkou Span verisi zaten bir kaydırma değ
//--- Chinkou Span çizgisine bir kaydırma değeri ayarlanması gerekmiyor
   if(CopyBuffer(ind_handle,4,0,amount,chinkou_span_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("5. iIchimoku göstergesinin verileri kopyalanamadı, hata kodu %d",Ge
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
```

```
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iBWMFI

Fonksiyon, Market Facilitation Index göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iBWMFI(
   string               symbol,             // sembol ismi
   ENUM_TIMEFRAMES      period,             // periyot
   ENUM_APPLIED_VOLUME  applied_volume      // hesaplamada kullanılacak hacim tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

applied_volume

[in] Kullanılan hacim. ENUM_APPLIED_VOLUME sabitlerinden biri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iBWMFI.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- iBWMFI grafiği
#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
//+------------------------------------------------------------------+
//|  İşleyici oluşturma yöntemleri için bir sayım                    |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iBWMFI,             // iBWMFI kulan
   Call_IndicatorCreate     // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iBWMFI;          // fonksiyon tipi
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;// hacim tipi
input string                symbol=" ";                // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // zaman aralığı
//--- gösterge tamponu
double          iBWMFIBuffer[];
double          iBWMFIColors[];
//--- iBWMFI göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Market Facilitation Index by Bill Williams göstergesindeki değerlerin sayısını t
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iBWMFIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iBWMFIColors,INDICATOR_COLOR_INDEX);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iBWMFI)
```

```
      handle=iBWMFI(name,period,applied_volume);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[1];
      //--- hacim tipi
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_BWMFI,1,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iBWMFI göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturula
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Market Facilitation Index by Bill Williams göstergesinin hesaplandığı sembol ve
   short_name=StringFormat("iBWMFI(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iBWMFI göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
```

```
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iBWMFI göstergesinin değerle
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iBWMFIBuffer dizisinin büyüklüğü iBWMFI gös
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- dizileri, Market Facilitation Index by Bill Williams göstergesinin değerleriyle
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArraysFromBuffers(iBWMFIBuffer,iBWMFIColors,handle,values_to_copy)) return
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Market Facilitation Index by Bill Williams göstergesindeki değerlerin sayısını h
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iBWMFI göstergesi ile, gösterge tamponlarının doldurulması       |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &values[],    // histogram değerleri için gösterge t
                           double &colors[],    // histogram renkleri için gösterge ta
                           int ind_handle,      // iBWMFI göstergesinin tanıtıcı değer
                           int amount           // kopyalanan değerlerin sayısı
                           )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iBWMFIBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iBWMFI göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastE
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
```

```
      return(false);
     }
//--- iBWMFIColors dizisinin bir kısmını 1 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,1,0,amount,colors)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iBWMFI göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastE
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iMomentum

Fonksiyon, Momentum göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iMomentum(
   string                symbol,            // sembol ismi
   ENUM_TIMEFRAMES       period,            // periyot
   int                   mom_period,        // ortalama periyodu
   ENUM_APPLIED_PRICE    applied_price      // fiyat veya işleyici tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

mom_period

[in] Fiyat değişimlerinin hesaplanması için gereken ortalama periyodu (çubuk sayısı olarak).

applied_price

[in] Kullanılan fiyat. ENUM_APPLIED_PRICE sayımındaki fiyat sabitlerinden biri veya başka bir göstergenin tanıtıcı değeri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                Demo_iMomentum.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla
#property description "Diğer tüm parametreler standart Momentum'daki gibidir."
```

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iMomentum grafiği
#property indicator_label1  "iMomentum"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMomentum,          // iMomentum kullan
   Call_IndicatorCreate     // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iMomentum;       // fonksiyon tipi
input int                   mom_period=14;             // Momentum periyodu
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // fiyat tipi
input string                symbol=" ";                // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // zaman aralığı
//--- gösterge tamponu
double          iMomentumBuffer[];
//--- iMomentum göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Momentum göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iMomentumBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
```

```
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iMomentum)
      handle=iMomentum(name,period,mom_period,applied_price);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[2];
      //--- periyot
      pars[0].type=TYPE_INT;
      pars[0].integer_value=mom_period;
      //--- fiyat tipi
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MOMENTUM,2,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iMomentum göstergesinin tanıtıcı değeri, %s/%s sembolü için oluştu
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Momentum göstergesinin hesaplandığı sembol ve zaman aralığı değerlerini göster
   short_name=StringFormat("iMomentum(%s/%s, %d, %s)",name,EnumToString(period),
                           mom_period, EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
```

```
//--- iMomentum göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iMomentum göstergesinin değe
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iMomentumBuffer dizisinin büyüklüğü iMoment
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iMomentumBuffer dizisini Momentum göstergesinin değerleri ile doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iMomentumBuffer,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Momentum göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iMomentum göstergesi ile, gösterge tamponlarının doldurulması    |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // Momentum değerleri için gösterge tampon
                         int ind_handle,    // iMomentum göstergesinin tanıtıcı değeri
                         int amount         // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
```

```
//--- iMomentumBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iMomentum göstergesinin verileri kopyalanamadı, hata kodu %d",GetLa
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iMFI

Fonksiyon, Money Flow Index göstergesinin tanıtıcı değerine dönüş yapar.

```
int  iMFI(
   string                symbol,              // sembol ismi
   ENUM_TIMEFRAMES       period,              // periyot
   int                   ma_period,           // ortalama periyodu
   ENUM_APPLIED_VOLUME   applied_volume       // hesaplamada kullanılacak hacim tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

ma_period

[in] Hesaplamalar için gereken ortalama periyodu (çubuk sayısı olarak).

applied_volume

[in] Kullanılan hacim. ENUM_APPLIED_VOLUME değerlerinden biri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iMFI.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla
#property description "Diğer tüm parametreler standart Money Flow Index göstergesinde

#property indicator_separate_window
```

```
#property indicator_buffers 1
#property indicator_plots   1
//--- iMFI grafiği
#property indicator_label1  "iMFI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- gösterge penceresindeki yatay seviyeler
#property indicator_level1  20
#property indicator_level2  80
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMFI,              // iMFI kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iMFI;               // fonksiyon tipi
input int                   ma_period=14;                 // periyot
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // hacim tipi
input string                symbol=" ";                   // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // zaman aralığı
//--- gösterge tamponu
double          iMFIBuffer[];
//--- iMFI göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Money Flow Index göstergesindeki değerlerin sayısını akılda tutacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iMFIBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
```

```
      {
       //--- göstergenin tutturulduğu çizelge sembolünü al
       name=_Symbol;
      }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iMFI)
      handle=iMFI(name,period,ma_period,applied_volume);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[2];
      //--- periyot
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- hacim tipi
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_MFI,2,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iMFI göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturulama
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Money Flow Index göstergesinin hesaplandığı sembol ve zaman aralığını göster
   short_name=StringFormat("iMFI(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period, EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iMFI göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iMFI göstergesinin değerleri
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iMFIBuffer dizisinin büyüklüğü iMFI gösterg
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iMFIBuffer dzisini Money Flow Index göstergesinin değerleriyle doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iMFIBuffer,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Money Flow Index göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iMFI göstergesi ile, gösterge tamponlarının doldurulması         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],  // Money Flow Index değerleri için gösterg
                         int ind_handle,    // iMFI göstergesinin tanıtıcı değeri
                         int amount         // kopyalanan değerlerin sayısı
                         )
  {
```

```
//--- hata kodunu sıfırla
   ResetLastError();
//--- iMFIBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iMFI göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastEr
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iMA

Fonksiyon, Hareketli Ortalama (Moving Average) göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iMA(
   string                symbol,            // sembol ismi
   ENUM_TIMEFRAMES       period,            // periyot
   int                   ma_period,         // ortalama periyodu
   int                   ma_shift,          // yatay kaydırma değeri
   ENUM_MA_METHOD        ma_method,         // düzleştirme tipi
   ENUM_APPLIED_PRICE    applied_price      // fiyat veya işleyici tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

ma_period

[in] Hareketli ortalamanın hesaplanması için ortalama periyodu.

ma_shift

[in] Fiyat çizelgesine göre, göstergenin kaydırma değeri.

ma_method

[in] Düzleştirme tipi. ENUM_MA_METHOD değerlerinden biri olabilir.

applied_price

[in] Kullanılan fiyat. ENUM_APPLIED_PRICE sayımındaki fiyat sabitlerinden biri veya başka bir göstergenin tanıtıcı değeri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                     Demo_iMA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
```

```
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla
#property description "Diğer tüm parametreler standart Hareketli ortalamadaki gibidir.

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iMA grafiği
#property indicator_label1  "iMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMA,               // iMA kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation             type=Call_iMA;               // fonksiyon tipi
input int                  ma_period=10;                // hareketli ortalama periyo
input int                  ma_shift=0;                  // kaydırma
input ENUM_MA_METHOD       ma_method=MODE_SMA;          // düzleştirme tipi
input ENUM_APPLIED_PRICE   applied_price=PRICE_CLOSE;   // fiyat tipi
input string               symbol=" ";                  // sembol
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;       // zaman aralığı
//--- gösterge tamponu
double         iMABuffer[];
//--- iMA göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Moving Average göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
```

```
   SetIndexBuffer(0,iMABuffer,INDICATOR_DATA);
//--- kaydırma değerini ayarla
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iMA)
      handle=iMA(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[4];
      //--- periyot
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- kaydırma değeri
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- düzleştirme tipi
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- fiyat tipi
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MA,4,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iMA göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturulamad
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Moving Average göstergesinin hesaplandığı sembol ve zaman aralığı değerlerini gö
   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);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iMA göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iMA göstergesinin değerleri
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iMABuffer dizisinin büyüklüğü iMA gösterges
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- iMABuffer dizisini Moving Average göstergesinin değerleriyle doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
```

```
                              values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Moving Average göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| MA göstergesi ile, gösterge tamponlarının doldurulması           |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &values[],   // Hareketli ortalama değerleri için göst
                         int shift,          // kaydırma değeri
                         int ind_handle,     // iMA göstergesinin tanıtıcı değeri
                         int amount          // kopyalanan değerler
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iMABuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,-shift,amount,values)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iMA göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastErro
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iOsMA

Fonksiyon, Moving Average of Oscillator göstergesinin tanıtıcı değerine dönüş yapar. OsMA osilatörü, MACD ve onun sinyal çizgisi değerlerinin arasındaki farkı gösterir. Tek bir tamponu vardır.

```
int  iOsMA(
   string               symbol,              // sembol ismi
   ENUM_TIMEFRAMES      period,              // periyot
   int                  fast_ema_period,     // Hızlı Hareketli Ortalama periyodu
   int                  slow_ema_period,     // Yavaş Hızlı Hareketli Ortalama periyodu
   int                  signal_period,       // fark için ortalama periyodu
   ENUM_APPLIED_PRICE   applied_price        // fiyat veya işleyici tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

fast_ema_period

[in] Hızlı Hareketli Ortalama periyodu.

slow_ema_period

[in] Yavaş Hareketli Ortalama periyodu.

signal_period

[in] sinyal çizgisinin hesabı için ortalama periyodu.

applied_price

[in] Kullanılan fiyat. ENUM_APPLIED_PRICE sayımındaki fiyat sabitlerinden biri veya başka bir göstergenin tanıtıcı değeri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Not

Bazı sistemlerde bu osilatör, MACD histogram olarak bilinir.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iOsMA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
```

```
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla
#property description "Diğer tüm parametreler standart Moving Average of Oscillator gö

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iOsMA grafiği
#property indicator_label1  "iOsMA"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iOsMA,              // iOsMA kullan
   Call_IndicatorCreate     // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iOsMA;           // fonksiyon tipi
input int                   fast_ema_period=12;        // hızlı hareketli ortalama peri
input int                   slow_ema_period=26;        // yavaş hareketli ortalama peri
input int                   signal_period=9;           // ortalamalar farkı periyodu
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // fiyat tipi
input string                symbol=" ";                // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // zaman aralığı
//--- gösterge tamponu
double         iOsMABuffer[];
//--- iAMA göstergesinin tanıtıcı değerini depolamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Moving Average göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
```

```
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iOsMABuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iOsMA)
      handle=iOsMA(name,period,fast_ema_period,slow_ema_period,signal_period,applied_p
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[4];
      //--- hızlı HO periyodu
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- yavaş HO periyodu
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- hızlı ve yavaş ortalamaların fark periyodu
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- fiyat tipi
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_OSMA,4,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iOsMA göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturular
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Moving Average of Oscillator göstergesinin hesaplandığı sembol ve zaman aralığı
```

```
   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);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iOsMA göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iOsMA göstergesinin değerler
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iOsMABuffer dizisinin büyüklüğü iOsMA göste
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- dizileri Alligator göstergesinin değerleriyle doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iOsMABuffer,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
```

```
                              TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                              short_name,
                              values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Moving Average of Oscillator göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iOsMA göstergesi ile, gösterge tamponlarının doldurulması        |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &ama_buffer[],  // OsMA değerleri için gösterge tampon
                         int ind_handle,        // iOsMA göstergesinin tanıtıcı değeri
                         int amount             // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iOsMABuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,ama_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iOsMA göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastEr
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iMACD

Fonksiyon, Moving Averages Convergence/Divergence göstergesinin tanıtıcı değerine dönüş yapar. OsMA'nın MACD Histogram olarak çağrıldığı sistemlerde, bu gösterge, iki çizgi ile gösterilir. Moving Averages Convergence/Divergence müşteri terminalinde bir histogram şeklinde gözükür.

```
int  iMACD(
   string               symbol,              // sembol ismi
   ENUM_TIMEFRAMES      period,              // periyot
   int                  fast_ema_period,     // hızlı ortalama periyodu
   int                  slow_ema_period,     // yavaş ortalama periyodu
   int                  signal_period,       // ortalamalar farkı
   ENUM_APPLIED_PRICE   applied_price        // fiyat veya işleyici tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

fast_ema_period

[in] Hızlı Hareketli Ortalama periyodu.

slow_ema_period

[in] Yavaş Hareketli Ortalama periyodu.

signal_period

[in] Sinyal çizgisinin hesaplanması için gereken periyot.

applied_price

[in] Kullanılan fiyat. ENUM_APPLIED_PRICE sayımındaki fiyat sabitlerinden biri veya başka bir göstergenin tanıtıcı değeri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Not

Tampon numaraları şu şekildedir: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                 Demo_iMACD.mq5 |
```

```
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla
#property description "diğer tüm parametreler, standart MACD gibidir."

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- MACD grafiği
#property indicator_label1  "MACD"
#property indicator_type1   DRAW_HISTOGRAM
#property indicator_color1  clrSilver
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Signal grafiği
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_DOT
#property indicator_width2  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iMACD,              // iMACD kullan
   Call_IndicatorCreate     // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iMACD;           // fonksiyon tipi
input int                   fast_ema_period=12;        // hızlı hareketli ortalama peri
input int                   slow_ema_period=26;        // yavaş hareketli ortalama peri
input int                   signal_period=9;           // ortalamalar farkı periyodu
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // fiyat tipi
input string                symbol=" ";                // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // zaman aralığı
//--- gösterge tamponları
double         MACDBuffer[];
double         SignalBuffer[];
//--- iMACD göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
```

```
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Moving Averages Convergence/Divergence göstergesindeki değerlerin sayısını koruy
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,MACDBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iMACD)
      handle=iMACD(name,period,fast_ema_period,slow_ema_period,signal_period,applied_p
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[4];
      //--- hızlı HO periyodu
      pars[0].type=TYPE_INT;
      pars[0].integer_value=fast_ema_period;
      //--- yavaş HO periyodu
      pars[1].type=TYPE_INT;
      pars[1].integer_value=slow_ema_period;
      //--- hızlı ve yavaş ortalamaların fark periyodu
      pars[2].type=TYPE_INT;
      pars[2].integer_value=signal_period;
      //--- fiyat tipi
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_MACD,4,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
```

```
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iMACD göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturular
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Moving Average Convergence/Divergence göstergesinin hesaplandığı sembol ve zaman
   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);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iMACD göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iMACD göstergesinin değerler
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için MACDBuffer dizisinin büyüklüğü iMACD göster
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
```

```
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- diziyi iMACD göstergesinin değerleriyle doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabili
   if(!FillArraysFromBuffers(MACDBuffer,SignalBuffer,handle,values_to_copy)) return(0)
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Moving Averages indicator Convergence/Divergence göstergesindeki değerlerin sayı
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iMACD göstergesi ile, gösterge tamponlarının doldurulması        |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &macd_buffer[],    // MACD değerleri için gösterge t
                           double &signal_buffer[],  // MACD sinyal çizgisi için göste
                           int ind_handle,           // iMACD göstergesinin tanıtıcı d
                           int amount                // kopyalanan değerlerin sayısı
                           )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iMACDBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,macd_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iMACD göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastEr
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }

//--- SignalBuffer dizisinin bir kısmını 1 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iMACD göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastEr
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
```

```
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iOBV

Fonksiyon, On Balance Volume gösterge tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iOBV(
   string                symbol,             // sembol ismi
   ENUM_TIMEFRAMES       period,             // periyot
   ENUM_APPLIED_VOLUME   applied_volume      // hesaplama için hacim tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

applied_volume

[in] Kullanılan hacim. ENUM_APPLIED_VOLUME değerlerinden biri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iOBV.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//---  iOBV
#property indicator_label1  "iOBV"
#property indicator_type1   DRAW_LINE
```

```
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iOBV ,              // iOBV kullan
   Call_IndicatorCreate     // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iOBV;               // fonksiyon tipi
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // hacim tipi
input string                symbol=" ";                   // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // zaman aralığı
//--- gösterge tamponları
double          iOBVBuffer[];
//--- iOBV göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- On Balance Volume göstergesindeki değerlerin sayısını saklayacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iOBVBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iOBV)
      handle=iOBV(name,period,applied_volume);
   else
     {
```

```
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[1];
      //--- hacim tipi
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_OBV,1,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iOBV göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturulama
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- On Balance Volume göstergesinin hesaplandığı sembol ve zaman aralığı değerlerini
   short_name=StringFormat("iOBV(%s/%s, %s)",name,EnumToString(period),
                           EnumToString(applied_volume));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iOBV göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iOBV göstergesinin değerleri
```

```
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iOBVBuffer dizisinin büyüklüğü iOBV gösterg
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- dizileri Alligator göstergesinin değerleriyle doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iOBVBuffer,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- On Balance Volume göstergesindeki değerlerin sayısını saklayacağız
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iOBV göstergesi ile, gösterge tamponlarının doldurulması         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &obv_buffer[],  // OBV değerleri için gösterge tamponu
                         int ind_handle,        // iOBV göstergesinin tanıtıcı değeri
                         int amount             // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iOBVBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,obv_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iOBV göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastErr
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
```

```
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iSAR

Fonksiyon, Parabolic Stop and Reverse system göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iSAR(
   string            symbol,      // sembol ismi
   ENUM_TIMEFRAMES   period,      // periyot
   double            step,        // artırma adımı
   double            maximum      // maksimal stop (durdurma) seviyesi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

step

[in] Durdurma seviyesi artırma değeri, genellikle 0.02.

maximum

[in] maksimal durdurma seviyesi, genellikle 0.2.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iSAR.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla
#property description "Diğer tüm parametreler standart Parabolic Stop and Reverse göst
```

```
#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iSAR çizimi
#property indicator_label1  "iSAR"
#property indicator_type1   DRAW_ARROW
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iSAR,              // iSAR kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation             type=Call_iSAR;               // fonksiyon tipi
input double               step=0.02;                    // adım değeri, stop seviyele
input double               maximum=0.2;                  // maksimum adım değeri
input string               symbol=" ";                   // sembol
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;        // zaman aralığı
//--- gösterge tamponları
double         iSARBuffer[];
//--- iSAR göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Parabolic SAR göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iSARBuffer,INDICATOR_DATA);
//--- Wingdings karakter kümesindeki sembolleri kullanarak PLOT_ARROW özelliği için k
   PlotIndexSetInteger(0,PLOT_ARROW,159);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
```

```
      {
       //--- göstergenin tutturulduğu çizelge sembolünü al
       name=_Symbol;
      }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iSAR)
      handle=iSAR(name,period,step,maximum);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[2];
      //--- adım değeri
      pars[0].type=TYPE_DOUBLE;
      pars[0].double_value=step;
      //--- hesaplamada kullanılan adım değerini sınırla
      pars[1].type=TYPE_DOUBLE;
      pars[1].double_value=maximum;
      handle=IndicatorCreate(name,period,IND_SAR,2,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iSAR göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturulama
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Parabolic SAR göstergesinin hesaplandığı sembol ve zaman aralığı değerlerini gös
   short_name=StringFormat("iSAR(%s/%s, %G, %G)",name,EnumToString(period),
                           step,maximum);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iSAR göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iSAR göstergesinin değerleri
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iSARBuffer dizisinin büyüklüğü iSAR gösterg
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- dizileri iSAR göstergesinin değerleriyle doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iSARBuffer,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Parabolic SAR göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iSAR göstergesi ile, gösterge tamponlarının doldurulması         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &sar_buffer[],  // Parabolic SAR değerleri için göste
                         int ind_handle,        // iSAR göstergesinin tanıtıcı değeri
                         int amount             // kopyalanan değerlerin sayısı
                         )
  {
```

```
//--- hata kodunu sıfırla
   ResetLastError();
//--- iSARBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,sar_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iSAR göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastEr
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iRSI

Fonksiyon, Relative Strength Index göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iRSI(
   string               symbol,            // sembol ismi
   ENUM_TIMEFRAMES      period,            // periyot
   int                  ma_period,         // ortalama periyodu
   ENUM_APPLIED_PRICE   applied_price      // fiyat veya işleyici tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

ma_period

[in] RSI hesabı için ortalama periyodu.

applied_price

[in] Kullanılan fiyat. ENUM_APPLIED_PRICE sayımındaki fiyat sabitlerinden biri veya başka bir göstergenin tanıtıcı değeri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iRSI.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla
#property description "Diğer tüm parametreler standart Relative Strength Index gösterg
```

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iRSI çizimi
#property indicator_label1  "iRSI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrDodgerBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- gösterge penceresinde gösterilen değerler için sınırlar
#property indicator_maximum 100
#property indicator_minimum 0
//--- gösterge penceresindeki yatay seviyeler
#property indicator_level1  70.0
#property indicator_level2  30.0
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iRSI,              // iRSI kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iRSI;               // fonksiyon tipi
input int                   ma_period=14;                 // ortalama periyodu
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE;    // fiyat tipi
input string                symbol=" ";                   // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // zaman aralığı
//--- gösterge tamponu
double          iRSIBuffer[];
//--- iRSI göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Relative Strength Index göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iRSIBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
```

```
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iRSI)
      handle=iRSI(name,period,ma_period,applied_price);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[2];
      //--- hareketli ortalama periyodu
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- hesaplamada kullanılan adım değerini sınırla
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_RSI,2,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iRSI göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturulama
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Relative Strength Index göstergesinin hesaplandığı sembol ve zaman aralığını gös
   short_name=StringFormat("iRSI(%s/%s, %d, %d)",name,EnumToString(period),
                           ma_period,applied_price);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iRSI göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iRSI göstergesinin değerleri
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iRSIBuffer dizisinin büyüklüğü iRSI gösterg
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- dizileri iRSI göstergesinin değerleri ile doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iRSIBuffer,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Relative Strength Index göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iRSI göstergesi ile, gösterge tamponlarının doldurulması         |
//+------------------------------------------------------------------+
```

```
bool FillArrayFromBuffer(double &rsi_buffer[],  // Relative Strength Index değerleri i
                         int ind_handle,        // iRSI göstergesinin tanıtıcı değeri
                         int amount             // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iRSIBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,rsi_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iRSI göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastEr
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iRVI

Relative Vigor Index göstergesinin tanıtıcı değerine dönüş yapar.

```
int  iRVI(
   string           symbol,        // sembol ismi
   ENUM_TIMEFRAMES  period,        // periyot
   int              ma_period      // ortalama periyodu
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

ma_period

[in] RVI hesabı için ortalama periyodu.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Not

Tampon numaraları şu şekildedir: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                   Demo_iRVI.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla
#property description "Diğer tüm parametreler standart Relative Vigor Index göstergesi

#property indicator_separate_window
```

```
#property indicator_buffers 2
#property indicator_plots   2
//--- RVI grafiği
#property indicator_label1  "RVI"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Signal grafiği
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iRVI,              // iRVI kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation             type=Call_iRVI;              // fonksiyon tipi
input int                  ma_period=10;                // hesaplama periyodu
input string               symbol=" ";                  // sembol
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;       // zaman aralığı
//--- gösterge tamponları
double         RVIBuffer[];
double         SignalBuffer[];
//--- iRVI göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Relative Vigor Index göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,RVIBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
```

```
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iRVI)
      handle=iRVI(name,period,ma_period);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[1];
      //--- Hesaplama için K periyodu
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      handle=IndicatorCreate(name,period,IND_RVI,1,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iRVI göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturulama
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Relative Vigor Index göstergesinin hesaplandığı sembol ve zaman aralığını göste
   short_name=StringFormat("iRVI(%s/%s, %d, %d)",name,EnumToString(period),ma_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iRVI göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iRVI göstergesinin değerleri
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için RVIBuffer dizisinin büyüklüğü iRVI gösterge
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- dizileri iRVI göstergesinin değerleriyle doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(RVIBuffer,SignalBuffer,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Relative Vigor göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iRVI göstergesi ile, gösterge tamponlarının doldurulması         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &rvi_buffer[],     // Relative Vigor Index değerleri i
                         double &signal_buffer[],  // sinyal çizgisinin tamponu
                         int ind_handle,           // iRVI göstergesinin tanıtıcı değe
                         int amount                // kopyalanan değerlerin sayısı
                         )
```

```
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iRVIBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,rvi_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iRVI göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastEr
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- SignalBuffer dizisinin bir kısmını 1 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,1,0,amount,signal_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iRVI göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastEr
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iStdDev

Fonksiyon, Standard Deviation (standart sapma) göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iStdDev(
   string               symbol,            // sembol ismi
   ENUM_TIMEFRAMES      period,            // periyot
   int                  ma_period,         // ortalama periyodu
   int                  ma_shift,          // yatay kaydırma değeri
   ENUM_MA_METHOD       ma_method,         // düzleştirme tipi
   ENUM_APPLIED_PRICE   applied_price      // fiyat veya işleyici tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

ma_period

[in] Gösterge hesabı için kullanılacak ortalama periyodu.

ma_shift

[in] Fiyat çizelgesine göre, göstergenin kaydırma değeri.

ma_method

[in] Ortalama periyodu. ENUM_MA_METHOD değerlerinden biri olabilir.

applied_price

[in] Kullanılan fiyat. ENUM_APPLIED_PRICE sayımındaki fiyat sabitlerinden biri veya başka bir göstergenin tanıtıcı değeri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                Demo_iStdDev.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
```

```
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla
#property description "Diğer tüm parametreler Standard Deviation'da olduğu gibidir."

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iStdDev grafiği
#property indicator_label1  "iStdDev"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrMediumSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iStdDev,            // iStdDev kullan
   Call_IndicatorCreate     // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iStdDev;         // fonksiyon tipi
input int                   ma_period=20;              // ortalama periyodu
input int                   ma_shift=0;                // kaydırma değeri
input ENUM_MA_METHOD        ma_method=MODE_SMA;        // düzeltme tipi
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // fiyat tipi
input string                symbol=" ";                // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // zaman aralığı
//--- gösterge tamponu
double          iStdDevBuffer[];
//--- iStdDev göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Standard Deviation göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
```

```
   SetIndexBuffer(0,iStdDevBuffer,INDICATOR_DATA);
//--- kaydırma değerini ayarla
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iStdDev)
      handle=iStdDev(name,period,ma_period,ma_shift,ma_method,applied_price);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[4];
      //--- periyot
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- kaydırma değeri
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- düzleştirme tipi
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_method;
      //--- fiyat tipi
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_STDDEV,4,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iStdDev göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturul
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Standard Deviation göstergesinin hesaplandığı sembol ve zaman aralığı değerlerin
   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);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iStdDev göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iStdDev göstergesinin değerl
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iStdDevBuffer dizisinin büyüklüğü iStdDev g
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- diziyi Standard Deviation göstergesinin değerleri ile doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iStdDevBuffer,ma_shift,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
```

```
                              values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Standard Deviation göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iStdDev göstergesi ile, gösterge tamponlarının doldurulması      |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &std_buffer[],  // Standard Deviation çizgisinin göste
                         int std_shift,         // Standard Deviation çizgisinin kaydı
                         int ind_handle,        // iStdDev göstergesinin tanıtıcı değe
                         int amount             // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iStdDevBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,-std_shift,amount,std_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iStdDev göstergesinin verileri kopyalanamadı, hata kodu %d",GetLast
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iStochastic

Fonksiyon, Stochastic Oscillator göstergesinin tanıtıcı değerine dönüş yapar.

```
int  iStochastic(
   string           symbol,          // sembol ismi
   ENUM_TIMEFRAMES  period,          // periyot
   int              Kperiod,         // K-periyot (hesaplanacak çubuk sayısı)
   int              Dperiod,         // D-periyot (ilk düzleştirme periyodu)
   int              slowing,         // son düzleştirme
   ENUM_MA_METHOD   ma_method,       // düzleştirme tipi
   ENUM_STO_PRICE   price_field      // stokastik hesaplama yöntemi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

Kperiod

[in] %K çizgisinin hesabı için ortalama periyodu (çubuk sayısıyla).

Dperiod

[in] %D çizgisinin hesabı için ortalama periyodu (çubuk sayısıyla).

slowing

[in] Yavaşlama değeri.

ma_method

[in] Ortalama periyodu. ENUM_MA_METHOD değerlerinden biri olabilir.

price_field

[in] hesaplamalar için fiyat seçimi parametresi. ENUM_STO_PRICE değerlerinden biri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Not

tampon numaraları: 0 - MAIN_LINE, 1 - SIGNAL_LINE.

Örnek:

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

```
//|                                                Demo_iStochastic.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla
#property description "Diğer tüm parametreler standart Stochastic Oscillator'de olduğu

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   2
//--- Stochastic grafiği
#property indicator_label1  "Stochastic"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrLightSeaGreen
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- Signal grafiği
#property indicator_label2  "Signal"
#property indicator_type2   DRAW_LINE
#property indicator_color2  clrRed
#property indicator_style2  STYLE_SOLID
#property indicator_width2  1
//--- gösterge değerlerinin sınırlarını ayarla
#property indicator_minimum 0
#property indicator_maximum 100
//--- gösterge penceresindeki yatay seviyeler
#property indicator_level1  -100.0
#property indicator_level2  100.0
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iStochastic,       // iStochastic kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iStochastic;     // fonksiyon tipi
input int                   Kperiod=5;                 // K periyodu (hesaplanacak çubu
input int                   Dperiod=3;                 // K periyodu (ilk düzleştirme p
input int                   slowing=3;                 // son düzleştirme periyodu
input ENUM_MA_METHOD        ma_method=MODE_SMA;        // düzeltme tipi
```

```
input ENUM_STO_PRICE       price_field=STO_LOWHIGH;   // Stochastic hesaplama yöntemi
input string               symbol=" ";                // sembol
input ENUM_TIMEFRAMES      period=PERIOD_CURRENT;     // zaman aralığı
//--- gösterge tamponları
double         StochasticBuffer[];
double         SignalBuffer[];
//--- iStochastic göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Stochastic Oscillator göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,StochasticBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,SignalBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iStochastic)
      handle=iStochastic(name,period,Kperiod,Dperiod,slowing,ma_method,price_field);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[5];
      //--- Hesaplama için K periyodu
      pars[0].type=TYPE_INT;
      pars[0].integer_value=Kperiod;
      //--- ilk düzleştirme için D periyodu
      pars[1].type=TYPE_INT;
      pars[1].integer_value=Dperiod;
      //--- son düzleştirme için K periyodu
      pars[2].type=TYPE_INT;
      pars[2].integer_value=slowing;
```

```
      //--- düzleştirme tipi
      pars[3].type=TYPE_INT;
      pars[3].integer_value=ma_method;
      //--- Stochastic hesaplama yöntemi
      pars[4].type=TYPE_INT;
      pars[4].integer_value=price_field;
      handle=IndicatorCreate(name,period,IND_STOCHASTIC,5,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iStochastic göstergesinin tanıtıcı değeri, %s/%s sembolü için oluş
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Stochastic Oscillator göstergesinin hesaplandığı sembol ve zaman aralığı değerle
   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);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iStochastic göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
```

```
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iStochastic göstergesinin de
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için StochasticBuffer dizisinin büyüklüğü iStoch
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- dizileri iStochastic göstergesinin değerleriyle doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArraysFromBuffers(StochasticBuffer,SignalBuffer,handle,values_to_copy)) ret
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Stochastic Oscillator göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iStochastic göstergesi ile, gösterge tamponlarının doldurulması  |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &main_buffer[],    // Stochastic Oscillator değerler
                           double &signal_buffer[],  // sinyal çizgisinin tamponu
                           int ind_handle,           // iStochastic göstergesinin tan
                           int amount                // kopyalanan değerlerin sayısı
                           )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- StochasticBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,MAIN_LINE,0,amount,main_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iStochastic göstergesinin verileri kopyalanamadı, hata kodu %d",Get
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
```

```
//--- SignalBuffer dizisinin bir kısmını 1 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,SIGNAL_LINE,0,amount,signal_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iStochastic göstergesinin verileri kopyalanamadı, hata kodu %d",Get
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iTEMA

Fonksiyon, Triple Exponential Moving Average göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iTEMA(
   string               symbol,            // sembol ismi
   ENUM_TIMEFRAMES      period,            // periyot
   int                  ma_period,         // ortalama periyodu
   int                  ma_shift,          // yatay kaydırma değeri
   ENUM_APPLIED_PRICE   applied_price      // fiyat veya işleyici tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

ma_period

[in] Hesaplamalar için gereken ortalama periyodu (çubuk sayısıyla).

ma_shift

[in] Fiyat çizelgesine göre, göstergenin kaydırma değeri.

applied_price

[in] Kullanılan fiyat. ENUM_APPLIED_PRICE sayımındaki fiyat sabitlerinden biri veya başka bir göstergenin tanıtıcı değeri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iTEMA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
```

```
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla
#property description "Diğer tüm parametreler standart Triple Exponential Moving Avera

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iTEMA grafiği
#property indicator_label1  "iTEMA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iTEMA,             // iTEMA kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iTEMA;           // fonksiyon tipi
input int                   ma_period=14;              // ortalama periyodu
input int                   ma_shift=0;                // kaydırma değeri
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // fiyat tipi
input string                symbol=" ";                // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // zaman aralığı
//--- gösterge tamponu
double          iTEMABuffer[];
//--- iTEMA göstergesinin tanıtıcı değerinin saklanması için değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Triple Exponential Moving Average göstergesindeki değerlerin sayısını koruyacağ
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iTEMABuffer,INDICATOR_DATA);
//--- kaydırma değerini ayarla
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- göstergenin çizileceği sembolü ayarla
```

```
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iTEMA)
      handle=iTEMA(name,period,ma_period,ma_shift,applied_price);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[3];
      //--- periyot
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- kaydırma değeri
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ma_shift;
      //--- fiyat tipi
      pars[2].type=TYPE_INT;
      pars[2].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TEMA,3,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iTEMA göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturular
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Triple Exponential Moving Average göstergesinin hesaplandığı sembol ve zaman ara
   short_name=StringFormat("iTEMA(%s/%s, %d, %d, %s)",name,EnumToString(period),
                           ma_period,ma_shift,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
```

```
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[])
  {
//--- iTEMA göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iTEMA göstergesinin değerler
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iTEMABuffer dizisinin büyüklüğü iTEMA göste
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- diziyi Triple Exponential Moving Average göstergesinin değerleriyle doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iTEMABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Triple Exponential Moving Average göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
```

```
  }
//+------------------------------------------------------------------+
//| iTEMA göstergesi ile, gösterge tamponlarının doldurulması        |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &tema_buffer[], // Triple Exponential Moving Average d
                         int t_shift,           // çizgi için kaydırma değeri
                         int ind_handle,        // iTEMA göstergesinin tanıtıcı değeri
                         int amount             // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iTEMABuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,-t_shift,amount,tema_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iTEMA göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastEr
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iTriX

Triple Exponential Moving Averages Oscillator göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iTriX(
   string               symbol,            // sembol ismi
   ENUM_TIMEFRAMES      period,            // periyot
   int                  ma_period,         // ortalama periyodu
   ENUM_APPLIED_PRICE   applied_price      // fiyat veya işleyici tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

ma_period

[in] Hesaplamalar için gereken ortalama periyodu (çubuk sayısı olarak).

applied_price

[in] Kullanılan fiyat. ENUM_APPLIED_PRICE sayımındaki fiyat sabitlerinden biri veya başka bir göstergenin tanıtıcı değeri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                  Demo_iTriX.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla
```

```
#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iTriX grafiği
#property indicator_label1  "iTriX"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iTriX,              // iTriX
   Call_IndicatorCreate     // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iTriX;           // fonksiyon tipi
input int                   ma_period=14;              // periyot
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // fiyat tipi
input string                symbol=" ";                // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // zaman aralığı
//--- gösterge tamponu
double         iTriXBuffer[];
//--- iTriX göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Triple Exponential Moving Averages Oscillator göstergesindeki değerlerin sayısın
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iTriXBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
```

```
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iTriX)
      handle=iTriX(name,period,ma_period,applied_price);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[2];
      //--- periyot
      pars[0].type=TYPE_INT;
      pars[0].integer_value=ma_period;
      //--- fiyat tipi
      pars[1].type=TYPE_INT;
      pars[1].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_TRIX,2,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iTriX göstergesinin tanıtıcı değeri %s/%s sembolü için oluşturulama
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Triple Exponential Moving Averages Oscillator göstergesinin hesaplandığı sembol
   short_name=StringFormat("iTriX(%s/%s, %d, %s)",name,EnumToString(period),
                           ma_period,EnumToString(applied_price));
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
```

```
//--- iTriX göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iStdDev göstergesinin değerl
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iTriXBuffer dizisinin büyüklüğü iTriX göste
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- diziyi Triple Exponential Moving Averages Oscillator göstergesinin değerleriyle
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iTriXBuffer,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Triple Exponential Moving Averages Oscillator göstergesindeki değerlerin sayısın
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iTriX göstergesi ile, gösterge tamponlarının doldurulması        |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &trix_buffer[], // Triple Exponential Moving Averages
                         int ind_handle,        // iTriX göstergesinin tanıtıcı değeri
                         int amount             // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
```

```
//--- iTriXBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,trix_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iTriX göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastEr
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iWPR

Larry Williams' Percent Range işleyicisine dönüş yapar. Tek bir tamponu vardır.

```
int  iWPR(
   string           symbol,          // sembol ismi
   ENUM_TIMEFRAMES  period,          // periyot
   int              calc_period      // ortalama periyodu
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

calc_period

[in] Gösterge hesabı için periyot değeri (çubuk sayısıyla).

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                   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 "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_separate_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iWPR grafiği
#property indicator_label1  "iWPR"
#property indicator_type1   DRAW_LINE
```

```
#property indicator_color1  clrCyan
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//--- gösterge değerlerinin sınırlarını ayarla
#property indicator_minimum -100
#property indicator_maximum 0
//--- gösterge penceresindeki yatay seviyeler
#property indicator_level1  -20.0
#property indicator_level2  -80.0
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iWPR,              // iWPR kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iWPR;            // fonksiyon tipi
input int                   calc_period=14;            // periyot
input string                symbol=" ";                // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // zaman aralığı
//--- gösterge tamponu
double          iWPRBuffer[];
//--- iWPR göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Larry Williams' Percent Range göstergesindeki değerlerin sayısını saklayacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iWPRBuffer,INDICATOR_DATA);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
```

```
      }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iWPR)
      handle=iWPR(name,period,calc_period);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[1];
      //--- periyot
      pars[0].type=TYPE_INT;
      pars[0].integer_value=calc_period;
      handle=IndicatorCreate(name,period,IND_WPR,1,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iWPR göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturulama
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Williams' Percent Range göstergesinin hesaplandığı sembol ve zaman aralığı değe
   short_name=StringFormat("iWPR(%s/%s, %d)",name,EnumToString(period),calc_period);
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iWPR göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
```

```
      {
       PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
       return(0);
      }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iWPR göstergesinin değerleri
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iWPRBuffer dizisinin büyüklüğü iForce göste
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- diziyi, Williams' Percent Range göstergesinin değerleri ile doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iWPRBuffer,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Larry Williams' Percent Range göstergesindeki değerlerin sayısını saklayacağız
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iWPR göstergesi ile, gösterge tamponlarının doldurulması         |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &wpr_buffer[],  // Williams' Percent Range değerleri i
                         int ind_handle,        // iWPR göstergesinin tanıtıcı değeri
                         int amount             // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iWPRBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,wpr_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iWPR göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastEr
```

```
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iVIDyA

Variable Index Dynamic Average göstergesinin tanıtıcı değerine dönüş yapar. Tek bir tamponu vardır.

```
int  iVIDyA(
   string               symbol,            // sembol ismi
   ENUM_TIMEFRAMES      period,            // periyot
   int                  cmo_period,        // Chande Momentum periyodu
   int                  ema_period,        // EMA düzleştirme periyodu
   int                  ma_shift,          // fiyat çizelgesi üzerindeki yatay kaydırma
   ENUM_APPLIED_PRICE   applied_price      // fiyat veya işleyici tipi
   );
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

cmo_period

[in] Chande Momentum Oscillator hesabı için (çubuk sayısıyla) periyot değeri.

ema_period

[in] Düzleştirme faktörünün hesaplanması için EMA periodu (çubuk sayısıyla).

ma_shift

[in] Fiyat çizelgesine göre, göstergenin kaydırma değeri.

applied_price

[in] Kullanılan fiyat. ENUM_APPLIED_PRICE sayımındaki fiyat sabitlerinden biri veya başka bir göstergenin tanıtıcı değeri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                 Demo_iVIDyA.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla
#property description "Diğer tüm parametreler standart Variable Index Dynamic Average

#property indicator_chart_window
#property indicator_buffers 1
#property indicator_plots   1
//--- iVIDyA grafiği
#property indicator_label1  "iVIDyA"
#property indicator_type1   DRAW_LINE
#property indicator_color1  clrBlue
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iVIDyA,            // iVIDyA kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iVIDyA;          // fonksiyon tipi
input int                   cmo_period=15;             // Chande Momentum periyodu
input int                   ema_period=12;             // düzleştirme faktörünün periyo
input int                   ma_shift=0;                // kaydırma değeri
input ENUM_APPLIED_PRICE    applied_price=PRICE_CLOSE; // fiyat tipi
input string                symbol=" ";                // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;     // zaman aralığı
//--- gösterge tamponu
double         iVIDyABuffer[];
//--- iVIDyA göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Variable Index Dynamic Average göstergesindeki değerlerin sayısını koruyacağız
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iVIDyABuffer,INDICATOR_DATA);
```

```
//--- kaydırma değerini ayarla
   PlotIndexSetInteger(0,PLOT_SHIFT,ma_shift);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iVIDyA)
      handle=iVIDyA(name,period,cmo_period,ema_period,ma_shift,applied_price);
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[4];
      //--- Chande Momentum periyodu
      pars[0].type=TYPE_INT;
      pars[0].integer_value=cmo_period;
      //--- düzleştirme faktörünün periyodu
      pars[1].type=TYPE_INT;
      pars[1].integer_value=ema_period;
      //--- kaydırma değeri
      pars[2].type=TYPE_INT;
      pars[2].integer_value=ma_shift;
      //--- fiyat tipi
      pars[3].type=TYPE_INT;
      pars[3].integer_value=applied_price;
      handle=IndicatorCreate(name,period,IND_VIDYA,4,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iVIDyA göstergesinin tanıtıcı değeri, %s/%s sembolü için oluşturula
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Variable Index Dynamic Average göstergesinin hesaplandığı sembol ve zaman aralığ
   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);
```

```
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iVIDyA göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iVIDyA göstergesinin değerle
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iVIDyABuffer dizisinin büyüklüğü iVIDyA gös
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- diziyi Variable Index Dynamic Average göstergesinin değerleriyle doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArrayFromBuffer(iVIDyABuffer,ma_shift,handle,values_to_copy)) return(0);
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
```

```
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Variable Index Dynamic Average indicator
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iVIDyA göstergesi ile, gösterge tamponlarının doldurulması       |
//+------------------------------------------------------------------+
bool FillArrayFromBuffer(double &vidya_buffer[],// Variable Index Dynamic Average değe
                         int v_shift,           // kaydırma değeri
                         int ind_handle,        // iVIDyA göstergesi
                         int amount             // kopyalanan değerlerin sayısı
                         )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iVIDyABuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,-v_shift,amount,vidya_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iVIDyA göstergesinin verileri kopyalanamadı, hata kodu %d",GetLastE
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

iVolumes

Volumes göstergesinin tanıtıcı değerine dönüş yapar. Sadece bir tamponu bulunmaktadır.

```
int  iVolumes(
   string               symbol,              // sembol ismi
   ENUM_TIMEFRAMES      period,              // periyot
   ENUM_APPLIED_VOLUME  applied_volume       // hesaplamada kullanılacak hacim tipi
   )
```

Parametreler

symbol

[in] Menkul değerin sembol ismi, gösterge hesabında kullanılması gereken veri. NULL değeri mevcut sembol anlamına gelir.

period

[in] Periyot değeri, ENUM_TIMEFRAMES değerlerinden biri olabilir, 0 mevcut zaman aralığını belirtir.

applied_volume

[in] Kullanılan hacim tipi. ENUM_APPLIED_VOLUME değerlerinden biri olabilir.

Dönüş değeri

Belirtilen teknik göstergenin tanıtıcı değerine dönüş yapar, başarısızlık durumunda ise, INVALID_HANDLE değerine dönüş yapar. Bilgisayar belleği, kullanılmayan göstergelerden temizlenmelidir. Bunun için, göstergenin tanıtıcı değerinin parametre olarak geçirildiği IndicatorRelease() fonksiyonu kullanılır.

Örnek:

```
//+------------------------------------------------------------------+
//|                                                 Demo_iVolumes.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2000-2024, MetaQuotes Ltd."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Gösterge, iVolumes teknik göstergesi için tamponlardan"
#property description "nasıl veri alınacağını göstermektedir."
#property description "Göstergenin hesaplanmasında kullanılacak sembol ve periyot,"
#property description "symbol ve period parametreleri ile ayarlanır."
#property description "İşleyicinin oluşturulması yöntemi 'type' parametresi ile ayarla

#property indicator_separate_window
#property indicator_buffers 2
#property indicator_plots   1
//--- iVolumes grafiği
#property indicator_label1  "iVolumes"
#property indicator_type1   DRAW_COLOR_HISTOGRAM
```

```
#property indicator_color1  clrGreen, clrRed
#property indicator_style1  STYLE_SOLID
#property indicator_width1  1
//+------------------------------------------------------------------+
//| İşleyici oluşturma yöntemleri için bir sayım                     |
//+------------------------------------------------------------------+
enum Creation
  {
   Call_iVolumes,          // iVolumes kullan
   Call_IndicatorCreate    // IndicatorCreate kullan
  };
//--- giriş parametreleri
input Creation              type=Call_iVolumes;           // fonksiyon tipi
input ENUM_APPLIED_VOLUME   applied_volume=VOLUME_TICK;   // hacim tipi
input string                symbol=" ";                   // sembol
input ENUM_TIMEFRAMES       period=PERIOD_CURRENT;        // zaman aralığı
//--- gösterge tamponları
double          iVolumesBuffer[];
double          iVolumesColors[];
//--- iVolumes göstergesinin tanıtıcı değerini saklamak için bir değişken
int    handle;
//--- kayıt için değişken
string name=symbol;
//--- çizelge üzerindeki gösterge ismi
string short_name;
//--- Volumes göstergesinin tanıtıcı değerini saklamak için bir değişken
int    bars_calculated=0;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- dizilerin gösterge tamponlarına atanması
   SetIndexBuffer(0,iVolumesBuffer,INDICATOR_DATA);
   SetIndexBuffer(1,iVolumesColors,INDICATOR_COLOR_INDEX);
//--- göstergenin çizileceği sembolü ayarla
   name=symbol;
//--- sağa ve sola doğru olan boşlukları sil
   StringTrimRight(name);
   StringTrimLeft(name);
//--- sonuçta 'name' dizgisi sıfır uzunluğa sahipse
   if(StringLen(name)==0)
     {
      //--- göstergenin tutturulduğu çizelge sembolünü al
      name=_Symbol;
     }
//--- göstergenin tanıtıcı değerini oluştur
   if(type==Call_iVolumes)
      handle=iVolumes(name,period,applied_volume);
```

```
   else
     {
      //--- yapıyı gösterge parametreleriyle doldur
      MqlParam pars[1];
      //--- fiyat tipi
      pars[0].type=TYPE_INT;
      pars[0].integer_value=applied_volume;
      handle=IndicatorCreate(name,period,IND_VOLUMES,1,pars);
     }
//--- işleyici oluşturulmadıysa
   if(handle==INVALID_HANDLE)
     {
      //--- hatayı belirt ve hata kodunu al
      PrintFormat("iVolumes göstergesinin tanıtıcı değeri, %s/%s sembolü için oluştur
                  name,
                  EnumToString(period),
                  GetLastError());
      //--- gösterge erken durduruldu
      return(INIT_FAILED);
     }
//--- Volumes göstergesinin hesaplandığı sembol ve zaman aralığı değerlerini göster
   short_name=StringFormat("iVolumes(%s/%s, %s)",name,EnumToString(period),EnumToStrin
   IndicatorSetString(INDICATOR_SHORTNAME,short_name);
//--- göstergenin normal yolla başlatılması
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Custom indicator iteration function                              |
//+------------------------------------------------------------------+
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[])
  {
//--- iVolumes göstergesinden kopyalanan değerlerin sayısı
   int values_to_copy;
//--- göstergede hesaplanan değerlerin sayısını belirle
   int calculated=BarsCalculated(handle);
   if(calculated<=0)
     {
      PrintFormat("BarsCalculated() %d dönüşü yaptı, hata kodu %d",calculated,GetLastE
      return(0);
     }
```

```
//--- bu, gösterge değerlerinin ilk hesaplanması ise veya iVolumes göstergesinin değer
//--- veya göstergeyi iki veya daha fazla çubuk için hesaplamak gerekiyorsa (yani fiya
   if(prev_calculated==0 || calculated!=bars_calculated || rates_total>prev_calculated
     {
      //--- verilen sembol ve periyot için iVolumesBuffer dizisinin büyüklüğü iVolumes
      //--- Aksi durumda, gösterge tamponlarının büyüklüklerinden daha az kopyalama ya
      if(calculated>rates_total) values_to_copy=rates_total;
      else                       values_to_copy=calculated;
     }
   else
     {
      //--- Bu, gösterge değerlerinin daha önce hesaplandığı anlamına geliyor OnCalcul
      //--- hesaplama için bir çubuktan fazlası eklenmeyecek
      values_to_copy=(rates_total-prev_calculated)+1;
     }
//--- dizileri iVolumes göstergesinin değerleriyle doldur
//--- FillArraysFromBuffer fonksiyonu false dönüşü yaparsa, bilgi henüz kullanılabilir
   if(!FillArraysFromBuffers(iVolumesBuffer,iVolumesColors,handle,values_to_copy)) ret
//--- mesajı oluştur
   string comm=StringFormat("%s ==>  göstergedeki gncellenen değer %s: %d",
                            TimeToString(TimeCurrent(),TIME_DATE|TIME_SECONDS),
                            short_name,
                            values_to_copy);
//--- servis mesajını çizelgede göster
   Comment(comm);
//--- Volumes göstergesindeki değerlerin sayısını hatırla
   bars_calculated=calculated;
//--- bir sonraki çağrı için prev_calculated değerine dönüş yap
   return(rates_total);
  }
//+------------------------------------------------------------------+
//| iVolumes göstergesi ile, gösterge tamponlarının doldurulması     |
//+------------------------------------------------------------------+
bool FillArraysFromBuffers(double &volume_buffer[],   // Volumes değerleri için göste
                           double &color_buffer[],    // renkler için gösterge tamponu
                           int ind_handle,            // iVolumes göstergesinin tanıt
                           int amount                 // kopyalanan değerler
                           )
  {
//--- hata kodunu sıfırla
   ResetLastError();
//--- iVolumesBuffer dizisinin bir kısmını 0 indisli tamponun değerleriyle doldur
   if(CopyBuffer(ind_handle,0,0,amount,volume_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iVolumes göstergesinin verileri kopyalanamadı, hata kodu %d",GetLas
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
```

```
//--- iVolumesColors dizisinin bir kısmını 1 indisli tamponun değerleriyle doldur 1
   if(CopyBuffer(ind_handle,1,0,amount,color_buffer)<0)
     {
      //--- kopyalama başarısız ise hata kodu al
      PrintFormat("iVolumes göstergesinin verileri kopyalanamadı, hata kodu %d",GetLas
      //--- sıfır sonuç ile çık - gösterge hesaplanmamış sayılacak
      return(false);
     }
//--- herşey yolunda
   return(true);
  }
//+------------------------------------------------------------------+
//| Indicator deinitialization function                              |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
   if(handle!=INVALID_HANDLE)
      IndicatorRelease(handle);
//--- göstergeyi sildikten sonra çizelgeyi temizle
   Comment("");
  }
```

Optimizasyon Sonuçları ile Çalışma

Strateji sınayıcıdaki optimizasyon sonuçlarının işlenmesi için düzenleme fonksiyonları. Bu fonksiyonlar, Uzman Danışmanlardan ve betiklerden, sınama temsilcilerindeki optimizasyon sırasında çağrılabilir.

Strateji sınayıcıda bir Uzman Danışmanı çalıştırırken, basit tipleri veya basit yapıları kullanarak kendi veri dizinizi oluşturabilirsiniz (bu dizi, dizgileri, sınıf nesnelerini veya dinamik dizi nesnelerini içeremez). Bu veri seti, FrameAdd() fonksiyonu kullanılarak çerçeve (frame) şeklinde adlandırılan özel bir yapıya kaydedilebilir. Bir uzman Danışmanın optimizasyonu sırasında her bir sınama temsilcisi, terminale çerçevelerden oluşan bir seri gönderebilir. Tüm çerçeveler alındıkları sıraya göre, Uzman Danışmanın adına, terminal_dizini/MQL5/Files/Tester dizininde *.MQD dosyasına yazılırla. Bunlar, sınama temsilcilerinden alındıkları sıraya göre yazılırlar. Terminalde sınama temsilcisinden alınan bir çerçeve, TesterPass olayını oluşturur.

Çerçeveler, belirtilen isimdeki bir dosyada ve bilgisayar belleğinde saklanabilirler. MQL5 dilinde çerçevelerin sayısıyla ilgili herhangi bir kısıtlama bulunmamaktadır.

MQL5 Bulut Ağında bellek ve disk alanı sınırları

MQL5 Bulut Ağında yürütülen optimizasyonlar için şu sınırlama mevcuttur: Uzman Danışman, diske 4 GB'tan fazla bilgi yazmamalı veya 4 GB'tan fazla RAM kullanmamalıdır. Sınır aşılırsa ağ temsilcisi hesaplamayı doğru bir şekilde tamamlayamaz ve dolayısıyla sonucu alamazsınız. Ancak yine de hesaplamalara harcanan tüm süre için ücretlendirilirsiniz.

Her optimizasyon geçişinden bilgi almanız gerekiyorsa, çerçeveleri diske yazmadan gönderin. MQL5 Bulut Ağında hesaplamalar sırasında Uzman Danışmandaki dosya işlemlerini kullanmaktan kaçınmak için aşağıdaki kontrolü kullanabilirsiniz:

```
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(...);
   ...
  }
```

Fonksiyon	Eylem
FrameFirst	Çerçevenin okuma işaretçisini başlangıca taşır ve önceden ayarlanmış olan filtreyi siler
FrameFilter	Çerçeve okuma filtresini ayarlar ve işaretçiyi başlangıca taşır
FrameNext	Bir çerçeveyi okur ve işaretçiyi bir sonrakine taşır
FrameInputs	Çerçevenin şekillendirildiği giriş parametresini alır

Fonksiyon	Eylem
FrameAdd	Verilerle birlikte bir çerçeve ekler
ParameterGetRange	Bir Uzman Danışmanın sınayıcıdaki optimizasyonu sırasında girdi değişkeni için, veri aralığı ve değişim birimi hakkında bilgi alır
ParameterSetRange	Bir Uzman Danışmanın sınayıcıdaki optimizasyonu sırasında girdi değişkeninin kullanımını belirler: değer,değişim birimi (adım değeri), başlangıç değeri ve son değer

Ayrıca Bakınız

Sınama İstatistikleri, Çalışan bir MQL5 Programının özellikleri

FrameFirst

Çerçevenin okuma işaretçisini başlangıca taşır ve önceden ayarlanmış olan filtreyi sıfırlar.

```
bool  FrameFirst();
```

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

FrameFilter

Çerçeve okuma filtresini ayarlar ve işaretçiyi başlangıca taşır.

```
bool  FrameFilter(
   const string  name,          // Genele açık isim/etiket
   long          id             // Genele açık tanımlayıcı
   );
```

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

Not

İlk parametreye bir boş dizgi geçirilmişse, filtre sadece bir sayısal parametre ile çalışır, yani sadece belirtilen tanımlayıcıya sahip çerçeveler görüntülenecektir. Eğer ikinci parametre ULONG_MAX ise, sadece bir metin filtresi çalışacaktır.

FrameFilter("", ULONG_MAX) çağrısı FrameFirst() çagrısına eş değerdir, yani hiçbir filtre kullanılmaz.

FrameNext

Bir çerçeveyi okur ve işaretçiyi bir sonrakine taşır. Fonksiyonun iki çeşidi bulunmaktadır.

1. Tek bir sayısal değer almak için çağrılır

```
bool  FrameNext(
   ulong&   pass,      // Bir çerçeve eklendiği sırada, optimizasyondaki bir geçişin n
   string&  name,      // Genele açık isim/etiket
   long&    id,        // Genele tanımlayıcı
   double&  value      // Değer
   );
```

2. Bir çerçevenin verilerini almak için çağrılır

```
bool  FrameNext(
   ulong&   pass,      // Bir çerçeve eklendiği sırada, optimizasyondaki bir geçişin n
   string&  name,      // Genele açık isim/etiket
   long&    id,        // Genele tanımlayıcı
   double&  value,     // Değer
   void&    data[]     // Herhangi tipte bir dizi
   );
```

Parametreler

pass

[out] Optimizasyon sırasındaki sınayıcıdaki bir geçişin numarası.

name

[out] Tanımlayıcının ismi.

id

[out] Tanımlayıcının değeri.

value

[out] Bir sayısal değer.

data

[out] Herhangi tipteki bir dizi.

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

Not

İkinci çağrı versiyonunda, *data[]* dizisindeki alınan verileri doğru şekilde işlemelisiniz.

FrameInputs

Belirtilen geçiş numarasına sahip çerçevenin şekillendirildiği giriş parametreleri.

```
bool  FrameInputs(
   ulong    pass,                // Optimizasyon sırasındaki geçiş numarası
   string&  parameters[],        // "parameterN=valueN" şeklindeki dizgilerden oluşan
   uint&    parameters_count     // toplam parametre sayısı
   );
```

Parametreler

pass

[in] Optimizasyon sırasındaki sınayıcıdaki bir geçişin numarası.

parameters

[out] parametreleri ve isimlerin açıklamalarını içeren dizgilerden oluşan bir dizi

parameters_count

[out] *parameters[]* dizisindeki elemanların sayısı.

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

Not

Dizgilerin sayısını (*parameters_count*), *parameters[]*, dizisinden elde ettiğinizde, tüm kayıtları incelemek için bir döngü ayarlayabilirsiniz. Bu, belirtilen bir geçiş sayısı için, bir Uzman Danışmanın giriş parametrelerinin değerlerini bulmanıza yardımcı olacaktır

FrameAdd

Veri içeren bir çerçeve ekler. Fonksiyonun iki çeşidi bulunmaktadır.

1. Bir dosyadan veri ekler

```
bool  FrameAdd(
   const string  name,        // Genele açık isim/etiket
   long          id,          // Genele açık kimlik (tanımlayıcı)
   double        value,       // Değer
   const string  filename     // Veri dosyasının ismi
   );
```

2. Herhangi tipteki bir diziden veri ekler

```
bool  FrameAdd(
   const string  name,        // Genele açık isim/etiket
   long          id,          // Genele açık kimlik (tanımlayıcı)
   double        value,       // Değer
   const void&   data[]       // Herhangi tipte bir dizi
   );
```

Parametreler

name

[in] Genele açık çerçeve ismi. Bu isim, bir filtre oluşturmak için FrameFilter() içinde kullanılabilir.

id

[in] Çerçeve için genele açık bir tanımlayıcı. Bu isim, bir filtre oluşturmak için FrameFilter() içinde kullanılabilir.

value

[in] Çerçeveye yazılacak sayısal değer. Bir geçişin sonucunu iletmek için OnTester() fonksiyonundaki gibi kullanılır.

filename

[in] Çerçeveye eklenecek veriyi içeren dosyanın dosyanın ismi. Dosya şu klasörde yer almalıdır MQL5/Files.

data

[in] Çerçeveye yazılacak herhangi bit tipteki dizi. Referans ile geçirilir.

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata hakkında bilgi almak için GetLastError() fonksiyonunu çağırın.

ParameterGetRange

Bir Uzman Danışmanın sınayıcıdaki optimizasyonu sırasında girdi değişkeni için, veri aralığı ve değişim birimi hakkında bilgi alır. Fonksiyonun iki çeşidi vardır.

1. Tamsayı tipli giriş parametresi için veri alır

```
bool  ParameterGetRange(
   const string  name,           // parametre (girdi değişkeni) ismi
   bool&         enable,         // parametre optimizasyonu devrede
   long&         value,          // parametre değeri
   long&         start,          // başlangıç değeri
   long&         step,           // değişim birimi (adım)
   long&         stop            // son değer
   );
```

2. Reel tipli giriş parametresi için veri alır

```
bool  ParameterGetRange(
   const string  name,           // parametre (girdi değişkeni) ismi
   bool&         enable,         // parametre optimizasyonu devrede
   double&       value,          // parametre değeri
   double&       start,          // başlangıç değeri
   double&       step,           // değişim birimi
   double&       stop            // son değer
   );
```

Parametreler

name

[in] girdi değişkeni tanımlayıcısı. Bu değişkenler bir uygulamanın dışsal parametreleridir. Değerleri, bir çizelge üzerinde çalıştırıldıklarında veya tekil bir sınama sırasında belirlenebilir.

enable

[out] Bu parametrenin, Strateji Sınama aracındaki optimizasyon sırasında, değerlerin listelenmesi için kullanılabilirliğini belirleyen bayrak.

value

[out] Parametre değeri.

start

[out] Parametrenin optimizasyonda kullanılacak başlangıç değeri.

step

[out] Parametre değerlerinin listelenmesi sırasındaki değişim birimi.

stop

[out] Parametrenin optimizasyonda kullanılacak son değer.

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata hakkında bilgi almak için GetLastError() fonksiyonunu kullanın.

Not

Bu fonksiyon sadece OnTesterInit(), OnTesterPass() ve OnTesterDeinit() işleyicilerinden çağrılabilir. Bu, Strateji Sınama Aracında optimizasyon sırasında Uzman Danışmanın giriş parametrelerinin değerlerini ve değişim aralıklarını almak için tasarlanmıştır.

OnTesterInit() içerisinde çağrıldığı zaman, elde edilen veriler, herhangi bir girdi değişkeninin listelenmesi için gereken kuralların yeniden tanımlanması için ParameterSetRange() fonksiyonu ile birlikte kullanılabilir. Böylece, yeni Start, Stop ve Step değerleri ayarlanabilir ve giriş parametresi, Strateji Sınama aracının ayarlarına bakılmaksızın optimizasyon sürecinden çıkarılabilir. Bu, Uzman Danışmanın anahtar parametrelerinin dışındaki bazı parametreleri optimizasyondan dışlayarak, optimizasyon sürecinde giriş parametreleri alanını yönetebilmenizi sağlar.

Örnek:

```
#property description "ParameterGetRange() fonksiyonunun uygulama örneği için bir Uzma
#property description "Strateji Sınama aracının optimizasyon modunda çalıştırılmalıdır
//--- giriş parametreleri
input int                  Input1=1;
input double               Input2=2.0;
input bool                 Input3=false;
input ENUM_DAY_OF_WEEK     Input4=SUNDAY;

//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Sadece Strateji Sınama aracında çalıştırılmak üzere tasarlanmış bir Uzman Danışm
   if(!MQL5InfoInteger(MQL5_OPTIMIZATION))
     {
      MessageBox("Strateji Sınama aracının optimizasyon modunda çalıştırılmalıdır!");
      //--- Uzman Danışman işlemini önceden sonlandır ve çizelgeden kaldır
      return(INIT_FAILED);
     }
//--- Başlatma başarıyla tamamlandı
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| TesterInit function                                              |
//+------------------------------------------------------------------+
void OnTesterInit()
  {
//--- long tipi giriş parametresi örneği
   string name="Input1";
   bool enable;
   long par1,par1_start,par1_step,par1_stop;
   ParameterGetRange(name,enable,par1,par1_start,par1_step,par1_stop);
```

```
   Print("İlk parametre");
   PrintFormat("%s=%d  enable=%s  den %d to %d'ye, adım=%d",
               name,par1,(string)enable,par1_start,par1_stop,par1_step);
//--- double tipi giriş parametresi örneği
   name="Input2";
   double par2,par2_start,par2_step,par2_stop;
   ParameterGetRange(name,enable,par2,par2_start,par2_step,par2_stop);
   Print("İkinci parametre");
   PrintFormat("%s=%G  enable=%s  from %G den %G'ye, adım=%G",
               name,par2,(string)enable,par2_start,par2_stop,par2_step);

//--- bool tipi giriş parametresi örneği
   name="Input3";
   long par3,par3_start,par3_step,par3_stop;
   ParameterGetRange(name,enable,par3,par3_start,par3_step,par3_stop);
   Print("Üçüncü parametre");
   PrintFormat("%s=%s  enable=%s'den %s'ye %s",
               name,(string)par3,(string)enable,
               (string)par3_start,(string)par3_stop);
//--- sayım tipi giriş parametresi örneği
   name="Input4";
   long par4,par4_start,par4_step,par4_stop;
   ParameterGetRange(name,enable,par4,par4_start,par4_step,par4_stop);
   Print("Dördüncü parametre");
   PrintFormat("%s=%s  enable=%s'den %s'ye %s",
               name,EnumToString((ENUM_DAY_OF_WEEK)par4),(string)enable,
               EnumToString((ENUM_DAY_OF_WEEK)par4_start),
               EnumToString((ENUM_DAY_OF_WEEK)par4_stop));
  }
//+------------------------------------------------------------------+
//| TesterDeinit function                                            |
//+------------------------------------------------------------------+
void OnTesterDeinit()
  {
//--- bu mesaj optimizasyon tamamlandıktan sonra görüntülenecek
   Print(__FUNCTION__," Optimizasyon tamamlandı");
  }
```

ParameterSetRange

Bir Uzman Danışmanın Strateji Sınama aracındaki optimizasyonu sırasında girdi değişkeninin kullanımını belirler: değer, değişim birimi (adım değeri), başlangç değeri ve son değer. Fonksiyonun iki çeşidi vardır.

1. Tamsayı tipli giriş parametresi için değer belirler

```
bool  ParameterSetRange(
   const string  name,          // parametre (girdi değişkeni) ismi
   bool          enable,        // parametre optimizasyonu devrede
   long          value,         // parametre değeri
   long          start,         // başlangıç değeri
   long          step,          // değişim birimi (adım)
   long          stop           // son değer
   );
```

2. Reel tipli giriş parametresi için değer belirler

```
bool  ParameterSetRange(
   const string  name,          // parametre (girdi değişkeni) ismi
   bool          enable,        // parametre optimizasyonu devrede
   double        value,         // parametre değeri
   double        start,         // başlangıç değeri
   double        step,          // değişim birimi
   double        stop           // son değer
   );
```

Parametreler

name

[in] input veya sinput değişken tanımlayıcısı. Bu değişkenler bir uygulamanın dışsal parametreleridir. Değerleri programın çalıştırılması sırasında belirlenebilir

enable

[in] Strateji Sınama aracındaki optimizasyon sırasında, değerlerin listelemek için bu parametreyi devreye sok.

value

[in] Parametre değeri.

start

[in] Parametrenin optimizasyonda kullanılacak başlangıç değeri.

step

[in] Parametre değerlerinin listelenmesi sırasındaki değişim birimi.

stop

[in] Parametrenin optimizasyonda kullanılacak son değer.

Dönüş değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar. Hata hakkında bilgi almak için GetLastError() fonksiyonunu kullanın.

Not

Bu fonksiyon Strateji Sınama aracında optimizasyonun çalıştırılması sırasında, sadece OnTesterInit() işleyicisinden çağrılabilir. Parametre aralığını ve değişim birimini (adım değeri) belirlemek için tasarlanmıştır. Parametre, Strateji Sınama aracının ayarlarına bakılmaksızın optimizasyon sürecinden çıkarılabilir. Bu, aynı zamanda, optimizasyon süreci içinde sinput şekillendiricisi ile bildirimi yapılan değişkenlerin kullanılmasını sağlar.

ParameterSetRange() fonksiyonu, bir Uzman Danışmanın strateji sınayıcıdaki optimizasyonunu (giriş parametrelerini optimizasyon sürecinden çıkarıp ekleyerek ve istenen aralığı ve değişim adımını ayarlayarak) anahtar parametrelerin değerlerine bağlı olarak yönetebilmenizi sağlar.

Olay İşleyici Fonksiyonları

Bu grup, özel olaylar ve zamanlayıcı olayları ile çalışmak için fonksiyonlar içerir. Bu gruba ek olarak, ön-tanımlı olayların işlenmesi için bazı özel fonksiyonlar da mevcuttur.

Fonksiyon	Eylem
EventSetMillisecondTimer	Yüksek çözünürlüklü zamanlayıcının olay oluşturucusunu, 1 saniyeden az olan bir periyot ile mevcut çizelge için çalıştırır
EventSetTimer	Zamanlayıcı olayının oluşturucusunu, belirtilen periyotta mevcut çizelge için başlatır
EventKillTimer	Mevcut çizelgede, zamanlayıcı tarafından olayların oluşturulmasını durdurur
EventChartCustom	Belirtilen çizelge için özel bir olay oluşturur

Ayrıca Bakınız

Çizelge Olaylarının Tipleri

EventSetMillisecondTimer

Bu fonksiyon müşteri terminaline, zamanlayıcı olaylarının, söz konusu uzman danışman veya gösterge için, bir saniyeden daha küçük aralıklarla oluşturulması gerektiğini belirtir.

```
bool  EventSetMillisecondTimer(
   int  milliseconds      // milisaniye sayısı
   );
```

Parametreler

milliseconds

[in] Zamanlayıcı olaylarının frekansını tanımlayan milisaniyelerin sayısı.

Dönüş değeri

Başarılı sonuç durumunda 'true', aksi durumda 'false' değerine dönüş yapar. Bir hata kodu alabilmek için, GetLastError() fonksiyonu çağrılmalıdır.

Not

Bu özellik, yüksek çözünürlüklü zamanlayıcıları gerektiren, yani, zamanlayıcı olaylarının saniyenin altındaki aralıklarla alınması gerektiği durumlar için tasarlanmıştır. Eğer saniyeden uzun aralıklarla çalışan geleneksel bir zamanlayıcı sizin için yeterli ise, EventSetTimer() fonksiyonunu kullanın.

Genelde, zamanlayıcının periyodu düşürüldüğünde, zamanlayıcı olaylarının işleyicisi daha fazla çağrılacağından, sınama zamanı uzayacaktır. Gerçek zamanlı modda çalışırken, zamanlayıcı olayları donanım sınırlamalarına bağlı olarak, en düşük 10-16 milisaniyelik aralıklarla oluşturulabilir.

Genel olarak, bu fonksiyon, OnInit() fonksiyonu içerisinden veya bir sınıf yapıcısı içinden çağrılmalıdır. Zamanlayıcıdan gelen olayların işlenebilmesi için, Uzman Danışmanın veya göstergenin OnTimer() fonksiyonunu içermesi gerekir.

Her Uzman Danışman ve gösterge, kendisine has olan zamanlayıcı ile ve sadece bundan gelen olaylar ile çalışır. Zamanlayıcı, MQL5 programının sonlandırılması durumunda, EventKillTimer() fonksiyonu tarafından devre dışı bırakılmamış olsa dahi, zorla yok edilir.

Her program için yanızca bir zamanlayıcı kurulabilir Her bir MQL5 uygulaması ve çizelgesi, kendisine has olay kuyruğuna sahiptir. Yeni ulaşan olaylar bu kuyruğa yerleştirilir. Eğer kuyrukta zaten bir Timer olayı varsa veya bu olay işlenme aşamasındaysa, yeni Timer olayı MQL5 uygulamasının olay kuyruğuna eklenmez.

EventSetTimer

Bu fonksiyon, söz konusu uzman danışman veya gösterge için, zamanlayıcı olaylarının belirtilen aralıklarla oluşturulması gerektiğini müşteri terminaline iletir.

```
bool  EventSetTimer(
   int  seconds      // saniyelerin sayısı
   );
```

Parametreler

seconds

[in] Zamanlayıcı olayının oluşma frekansını belirleyen saniyelerin sayısı.

Dönüş değeri

Başarılı sonuç durumunda 'true', aksi durumda ise 'false' dönüşü yapar. Bir hata kodu almak için, GetLastError() fonksiyonunu çağırın.

Not

Normalde bu fonksiyon, OnInit() fonksiyonunun veya bir sınıf yapıcısı içinden çağrılmalıdır. Zamanlayıcıdan gelen olayların işlenebilmesi için, Uzman Danışmanın OnTimer() fonksiyonunu içermesi gerekir.

Tüm göstergeler ve Uzman Danışmanlar kendi saatleri (timer) ile çalışırlar ve olayları sadece bu saatten alırlar MQL5 programı çalışmayı durdurduğu anda zamanlayıcı, EventKillTimer() fonksiyonu ile devre dışı bırakılmamışsa zorla sonlandırılır.

Her program için sadece bir adet zamanlayıcı çalıştırılabilir. Her bir MQL5 uygulaması ve çizelgesi, kendilerine has olay kuyruğuna sahiptirler. Yeni ulaşan olaylar bu kuyruğa yerleştirilir. Eğer kuyrukta zaten bir Timer olayı varsa veya bu olay işlenme aşamasındaysa, yeni Timer olayı MQL5 uygulamasının olay kuyruğuna eklenmez.

EventKillTimer

Zamanlayıcıdan yapılan olayları sonlandırmak için gereken müşteri terminalini belirtir.

```
void  EventKillTimer();
```

Dönüş değeri

Dönüş değeri yok.

Not

EventSetTimer() fonksiyonu, OnInit() fonksiyonundan çağrılmışsa, sonlandırma fonksiyonunun OnDeinit() fonksiyonundan çağrılması gerekir. Benzer şekilde eğer EventSetTimer() fonksiyonu bir sınıf yapıcısı içinde çağrılmışsa, EventKillTimer() fonksiyonu bu sınıfın yıkıcısı içinde de çağrılabilir.

Tüm göstergeler ve Uzman Danışmanlar kendi saatleri (timer) ile çalışırlar ve olayları sadece bu saatten alırlar MQL5 programı çalışmayı durdurduğu anda zamanlayıcı, EventKillTimer() fonksiyonu ile devre dışı bırakılmamışsa zorla sonlandırılır

EventChartCustom

Bu fonksiyon, belirtilen çizelge için özel bir olay oluşturur.

```
bool  EventChartCustom(
   long    chart_id,            // olayın alındığı çizelgenin tanımlayıcısı
   ushort  custom_event_id,     // olay tanımlayıcı
   long    lparam,              // long tipli parametre
   double  dparam,              // double tipli parametre
   string  sparam               // string tipli olay parametresi
   );
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı. 0, mevcut çizelge anlamına gelir.

custom_event_id

[in] Kullanıcı tanımlı olaylar için tanımlayıcı. Bu tanımlayıcı otomatik olarak CHARTEVENT_CUSTOM değerine eklenir ve tamsayı tipine dönüştürür.

lparam

[in] OnChartEvent fonksiyonuna geçirilen long tipli olay parametresi.

dparam

[in] OnChartEvent fonksiyonuna geçirilen double tipli olay parametresi.

sparam

[in] OnChartEvent fonksiyonuna geçirilen string tipli olay parametresi. Eğer dizgi 63 karakterden uzunsa budanır.

Dönüş değeri

Eğer bir özel olay, çizelgenin olaylar kuyruğuna başarılı bir şekilde yerleştirilmişse, 'true' dönüşü yapar Başarısızlık durumunda ise 'false' dönüşü yapar. Bir hata kodu almak için GetLastError() fonksiyonunu kullanın.

Not

Belirtilen çizelgeye tutturulmuş Uzman Danışmanlar ve göstergeler, olayı işlemek için OnChartEvent(int event_id, long& lparam, double& dparam, string& sparam) fonksiyonunu kullanırlar.

Her olay tipi için, OnChartEvent() fonksiyonunun giriş parametreleri, mevcut olayı işleyebilmek için kesin değerlere sahiptir. Bu parametreler ile geçirilen olaylar ve değerler aşağıdaki tabloda listelenmiştir.

Olay	id parametresinin değeri	lparam parametresinin değeri	dparam parametresinin değeri	sparam parametresinin değeri
Tuş darbesi olayı	CHARTEVENT_KEYDOWN	basılmış bir tuşun kodu	Tekrar sayısı (kullanıcının tuşa	Klavye tuşlarının durumunu tarif

Olay	id parametresinin değeri	lparam parametresinin değeri	dparam parametresinin değeri	sparam parametresinin değeri
			basılı tutması sonucu tekrarlanan tuş darbelerinin sayısı)	eden bir bit maskesinin dizgi değeri
Fare olayı (çizelge için CHART_EVENT_MOUSE_MOVE=true şeklinde ayarlanmışsa)	CHARTEVENT_MOUSE_MOVE	X koordinatı	Y koordinatı	Fare tuşlarının durumunu tarif eden bir bit maskesinin string tipli değeri
Grafiksel nesne oluşturma olayı (çizelge için CHART_EVENT_OBJECT_CREATE=true şeklinde ayar yapılmışsa)	CHARTEVENT_OBJECT_CREATE	—	—	Oluşturulan grafiksel nesnenin adı
Özellikler iletişim kutusu ile bir nesnenin özelliğinin değiştirilmesi olayı	CHARTEVENT_OBJECT_CHANGE	—	—	Değiştirilen grafiksel nesnenin adı
Grafiksel nesnenin silinmesi olayı (çizelge için CHART_EVENT_OBJECT_DELETE=true şeklinde ayarlanmışsa)	CHARTEVENT_OBJECT_DELETE	—	—	Silinen grafiksel nesnenin ismi
Çizelge üzerinde fare tıklaması olayı	CHARTEVENT_CLICK	X koordinatı	Y koordinatı	—
Çizelge üzerindeki bir grafiksel nesnenin fare ile tıklanması olayı	CHARTEVENT_OBJECT_CLICK	X koordinatı	Y koordinatı	Olayın gerçekleştiği grafiksel nesnenin ismi
Grafiksel nesnenin fare ile	CHARTEVENT_OBJECT_DRAG	—	—	Taşınan grafiksel nesnenin adı

Olay	id parametresinin değeri	lparam parametresinin değeri	dparam parametresinin değeri	sparam parametresinin değeri
sürüklenmesi olayı				
LabelEdit grafiksel nesnesinin giriş kutusunda yapılan metin düzenleme işinin bitmesi olayı	CHARTEVENT_OBJECT_ENDEDIT	—	—	Metin düzenlemesi tamamlanan LabelEdit grafiksel nesnesinin adı
Bir çizelge üzerindeki değişimlerin olayı	CHARTEVENT_CHART_CHANGE	—	—	—
N sayısının altındaki kullanıcı olayının kimliği	CHARTEVENT_CUSTOM+N	EventChartCustom() fonksiyonu tarafından ayarlanmış değer	EventChartCustom() fonksiyonu tarafından ayarlanmış değer	EventChartCustom() fonksiyonu tarafından ayarlanmış değer

Örnek:

```
//+------------------------------------------------------------------+
//|                                              ButtonClickExpert.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                                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;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Özel olaylar göndermek için bir düğme oluştur
   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);

//--- Bilgiyi görüntülemek için bir etiket oluştur
   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);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   ObjectDelete(0,buttonID);
   ObjectDelete(0,labelID);
  }
//+------------------------------------------------------------------+
//| Expert tick function                                             |
//+------------------------------------------------------------------+
void OnTick()
  {
//---

  }
//+------------------------------------------------------------------+
void OnChartEvent(const int id,
                  const long &lparam,
                  const double &dparam,
                  const string &sparam)
  {
//--- Fareyi tıklayarak olayı kontrol et
   if(id==CHARTEVENT_OBJECT_CLICK)
     {
      string clickedChartObject=sparam;
      //--- buttonID isimli nesneye tıkladıysan
      if(clickedChartObject==buttonID)
        {
```

```
         //--- Düğme durumu - basılı veya serbest
         bool selected=ObjectGetInteger(0,buttonID,OBJPROP_STATE);
         //--- bir hata düzeltme mesajını günlüğe yaz
         Print("Düğmeye basıldı = ",selected);
         int customEventID; // Gönderilecek özel olayların sayısı
         string message;    // Olay ile gönderilecek mesaj
         //--- Eğer düğmeye basılmışsa
         if(selected)
           {
            message="Düğmeye basıldı";
            customEventID=CHARTEVENT_CUSTOM+1;
           }
         else // Düğmeye basılmamışsa
           {
            message="Düğmeye basıldı";
            customEventID=CHARTEVENT_CUSTOM+999;
           }
         //--- "our" isimli bir özel olay gönder
         EventChartCustom(0,customEventID-CHARTEVENT_CUSTOM,0,0,message);
         ///--- tüm açık çizelgelere bir mesaj gönder
         BroadcastEvent(ChartID(),0,"Broadcast Message");
         //--- Hata ayıklama mesajı
         Print("Bir olay gönderildi, ID = ",customEventID);
        }
      ChartRedraw();// Tüm çizelge nesnelerinin yeniden çizilmeye zorlanması
     }

//--- Olayın özel bir olay olup olmadığını kontrol et
   if(id>CHARTEVENT_CUSTOM)
     {
      if(id==broadcastEventID)
        {
         Print("Bir çizelgeden yayınlanan bir mesaj alındı, id = "+lparam);
        }
      else
        {
         //--- Olaydaki metin mesajını okuruz
         string info=sparam;
         Print("Kullanıcı olayını işle, ID = ",id);
         //--- Bir etiket içerisinde mesajı görüntüle
         ObjectSetString(0,labelID,OBJPROP_TEXT,sparam);
         ChartRedraw();// Tüm çizelge nesnelerinin yeniden çizilmeye zorlanması
        }
     }
  }
//+------------------------------------------------------------------+
//| tüm açık çizelgelere yayın olayı gönderir                        |
//+------------------------------------------------------------------+
void BroadcastEvent(long lparam,double dparam,string sparam)
```

```
  {
   int eventID=broadcastEventID-CHARTEVENT_CUSTOM;
   long currChart=ChartFirst();
   int i=0;
   while(i<CHARTS_MAX)                 // Kesinlikle CHARTS_MAX değerinden daha fazla
     {
      EventChartCustom(currChart,eventID,lparam,dparam,sparam);
      currChart=ChartNext(currChart); // Daha öncekilerden yeni bir çizelgemiz var
      if(currChart==-1) break;        // Çizelge listesinin sonuna gelindi
      i++;// Sayacı artırmayı unutma
     }
  }
//+------------------------------------------------------------------+
```

Ayrıca Bakınız

Müşteri terminali olayları, Olay işleyici fonksiyonlar

OpenCL ile Çalışma

OpenCL programları, OpenCL 1.1 (veya daha üstünü) destekleyen ekran kartları üzerinde hesaplamalar yapma amacıyla kullanılır. Modern ekran kartları, gelen veri akışıyla ilgi basit matematiksel işlemleri eş-anlı olarak gerçekleştirebilecek, özelleştirilmiş yüzlerce küçük işlemci içerir. OpenCL dili bu işlemcileri kullanmak amacıyla paralel hesaplamayı düzenler. Böylece bazı görevler için daha üstün bir işlem hızı elde etmemizi sağlar.

Bazı ekran kartlarında double tipli sayılar varsayılan olarak devre dışı bırakılmıştır. Bu 5105 kodlu derleme hatasına yol açabilir. double tipli sayı desteğini etkinleştirmek için, şu direktifi OpenCL programına ekleyin: #pragma OPENCL EXTENSION cl_khr_fp64 : enable. Ekran kartınız double tipini desteklemiyorsa bu direktif işe yaramaz.

OpenCL kaynak kodunun ayrı CL dosyalarına yazılması tavsiye edilir. Bunlar daha sonra kaynak değişkenleri yardımıyla MQL5 programına eklenebilir.

OpenCL programlarında hata işleme

Bir OpenCL programındaki son hata hakkında bilgi almak için, son hatanın kodunu ve açıklamasını geri döndüren CLGetInfoInteger ve CLGetInfoString fonksiyonlarını kullanın.

Son OpenCL hatasının kodu: Son OpenCL hatasının kodunu almak için, *handle* parametresini göz ardı ederek (sıfır belirtilebilir) CLGetInfoInteger'ı çağırın. Hataların açıklamaları: https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS.

Bilinmeyen bir hata kodu için "Bilinmeyen OpenCL hatası: N" dizgesi geri döndürülür, burada N hata kodudur. Örnek:

```
//--- son hatanın kodu alınırken handle parametresi göz ardı edilir
intcode= (int)CLGetInfoInteger(0,CL_LAST_ERROR);
```

Son OpenCL hatasının açıklaması: Son OpenCL hatasının açıklamasını almak için CLGetInfoString'i çağırın. Son OpenCL hatasının kodu *handle* parametresi olarak iletilmelidir.

Hataların açıklamaları: https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS. Hata kodu yerine CL_LAST_ERROR iletilirse, fonksiyon yine son hatanın açıklamasını geri döndürecektir. Örnek:

```
//--- son OpenCL hatasının kodunu al
int    code= (int)CLGetInfoInteger(0,CL_LAST_ERROR);
stringdesc;// hata açıklamasını almak için

//--- hata açıklamasını almak için hata kodunu kullan
if(!CLGetInfoString(code,CL_ERROR_DESCRIPTION,desc))
 desc="OpenCL hatasının açıklaması alınamıyor,"+ (string)GetLastError();
Print(desc);

//--- son OpenCL hatasının açıklamasını, son OpenCL hatasının kodu olmadan almak için
if(!CLGetInfoString(CL_LAST_ERROR,CL_ERROR_DESCRIPTION,desc))
 desc="OpenCL hatasının açıklaması alınamıyor,"+ (string)GetLastError();
Print(desc);;
```

Hata açıklaması olarak dahili numaralandırmanın adı geri döndürülür. Ayrıntılı açıklamaları buradan bulabilirsiniz: https://registry.khronos.org/OpenCL/specs/3.0-unified/html/OpenCL_API.html#CL_SUCCESS. Örneğin, CL_INVALID_KERNEL_ARGS değeri, "Bazı çekirdek argümanları ayarlanmadığında veya geçersiz olduğunda çekirdeği kuyruğa alırken geri döndürülür" anlamına gelir.

OpenCL ile çalışan programlar için fonksiyonlar:

Fonksiyon	Eylem
CLHandleType	Bir OpenCL tanıtıcı değerinin (handle) tipine, ENUM_OPENCL_HANDLE_TYPE sayımının değerlerinden biri şeklinde dönüş yapar
CLGetInfoInteger	Bir OpenCL aygıtı veya nesnesi için bir tamsayı özelliğinin değerine dönüş yapar
CLContextCreate	Bir OpenCL bağlamı oluşturur
CLContextFree	Bir OpenCL bağlamını siler
CLGetDeviceInfo	OpenCL sürücüsünden aygıtın özelliğini alır
CLProgramCreate	Bir kaynak kodundan OpenCL programı oluşturur
CLProgramFree	Bir OpenCL programını siler
CLKernelCreate	Bir OpenCL başlatma fonksiyonu oluşturur
CLKernelFree	OpenCL başlatma fonksiyonunu siler
CLSetKernelArg	OpenCL fonksiyonu için parametre ayarlar
CLSetKernelArgMem	Bir OpenCL tamponunu OpenCL fonksiyonunun parametresi olarak ayarlar
CLSetKernelArgMemLocal	Yerel tamponu kernel fonksiyonunun bir argümanı olarak ayarlar
CLBufferCreate	Bir OpenCL tamponu oluşturur
CLBufferFree	Bir OpenCL tamponunu siler
CLBufferWrite	Bir diziyi OpenCL tamponuna yazar
CLBufferRead	OpenCL tamponunu bir diziye okur
CLExecute	Bir OpenCL programını çalıştırır
CLExecutionStatus	OpenCL programının çalışma durumuna dönüş yapar

Ayrıca Bakınız

OpenCL, Kaynaklar

CLHandleType

Bir OpenCL tanıtıcı değerinin tipine, ENUM_OPENCL_HANDLE_TYPE sayımının değerlerinden biriyle dönüş yapar.

```
ENUM_OPENCL_HANDLE_TYPE  CLHandleType(
   int  handle     // Bir OpenCL nesnesinin tanıtıcı değeri
   );
```

Parametreler

handle

[in] Bir OpenCL nesnesinin (bir bağlam, bir kernel veya bir OpenCL programı) tanıtıcı değeri.

Dönüş değeri

ENUM_OPENCL_HANDLE_TYPE sayımının değerlerinden biri şeklinde, OpenCL tanıtıcı değerinin tipi.

ENUM_OPENCL_HANDLE_TYPE

Tanımlayıcı	Açıklama
OPENCL_INVALID	Hatalı tanıtıcı değer
OPENCL_CONTEXT	Bir OpenCL bağlamının tipi
OPENCL_PROGRAM	Bir OpenCL programının tipi
OPENCL_KERNEL	Bir OpenCL kernelinin tipi
OPENCL_BUFFER	Bir OpenCL tamponunun tipi

CLGetInfoInteger

Bir OpenCL nesnesi veya aygıtı için bir tamsayı özelliğinin değerine dönüş yapar.

```
long  CLGetInfoInteger(
   int  handle,                          // OpenCL nesnesinin tanıtıcı değeri veya Op
   ENUM_OPENCL_PROPERTY_INTEGER  prop    // İstenen özellik
   );
```

Parametreler

handle

[in] OpenCL nesnesinin tanıtıcı değeri veya OpenCL aygıtının numarası. OpenCL aygıtlarının numaralandırılmasına sıfır ile başlanır.

prop

[in] Alınması istenilen özellik değeri, ENUM_OPENCL_PROPERTY_INTEGER sayımının değerlerinden biri olabilir.

Dönüş değeri

Başarılı olunması durumunda özellik değerine, aksi durumda -1 durumuna dönüş yapılır. Hata hakkında bilgi almak için GetLastError() fonksiyonunu kullanın.

ENUM_OPENCL_PROPERTY_INTEGER

Tanımlayıcı	Açıklama	Tip
CL_DEVICE_COUNT	OpenCL desteğine sahip aygıtların sayısı. Bu özellik için ilk parametrenin belirtilmesi gerekmez, yani *handle* parametresinin yerine 0 değerini geçebilirsiniz.	int
CL_DEVICE_TYPE	Aygıt Tipi	ENUM_CL_DEVICE_TYPE
CL_DEVICE_VENDOR_ID	Tedarikçinin benzersiz kimliği	uint
CL_DEVICE_MAX_COMPUTE_UNITS	OpenCL aygıtında paralel olarak hesaplanmış görevlerin sayısı. Her bir çalışma grubu, sadece bir hesaplama görevini üstlenir. 1 en düşük değerdir	uint
CL_DEVICE_MAX_CLOCK_FREQUENCY	Aygıtın ayarlanmış en yüksek frekansı - MHz bazında.	uint
CL_DEVICE_GLOBAL_MEM_SIZE	Aygıtın global belleğinin büyüklüğü - bayt bazında	ulong
CL_DEVICE_LOCAL_MEM_SIZE	Yerel hafızada işlenen (scene) verinin büyüklüğü - bayt bazında	uint

Tanımlayıcı	Açıklama	Tip
CL_BUFFER_SIZE	OpenCL tamponunun tam boyutu (bayt olarak)	ulong
CL_DEVICE_MAX_WORK_GROUP_SIZE	OpenCL cihazı için kullanılabilir yerel çalışma gruplarının toplam sayısı.	ulong
CL_KERNEL_WORK_GROUP_SIZE	OpenCL programı için kullanılabilir yerel çalışma gruplarının toplam sayısı.	ulong
CL_KERNEL_LOCAL_MEM_SIZE	Bir gruptaki tüm paralel görevlerin çözümü için OpenCL programı tarafından kullanılan yerel belleğin boyutu (bayt olarak). Maksimum mevcut değeri almak için CL_DEVICE_LOCAL_MEM_SIZE kullan	ulong
CL_KERNEL_PRIVATE_MEM_SIZE	OpenCL program kernelindeki ger bir görev tarafından kullanılan özel belleğin minimum boyutu (bayt olarak).	ulong
CL_LAST_ERROR	Son OpenCL hatasının kodu	int

ENUM_CL_DEVICE_TYPE sayımı, OpenCL desteğine sahip mevcut aygıtların muhtemel tiplerini içerir. Aygıt tipini, aygıt numarası veya OpenCL nesnesinin tanıtıcı değeri ile elde edebilirsiniz; tanıtıcı değeri almak için CLGetInfoInteger(handle_or_deviceN, CL_DEVICE_TYPE) fonksiyonunu kullanabilirsiniz.

ENUM_CL_DEVICE_TYPE

Tanımlayıcı	Açıklama
CL_DEVICE_ACCELERATOR	Adanmış OpenCL hızlandırıcıları (örneğin, IBM CELL Blade). Bu aygıtlar, PCIe gibi bir periferal bağlantı kullanarak ana işlemci ile iletişim kurar.
CL_DEVICE_CPU	Bir OpenCL ana işlemci aygıtı. Ana işlemci OpenCL uygulamalarını çalıştırır, tek veya çok çekirdekli bir CPU olabilir.
CL_DEVICE_GPU	Bir OpenCL GPU (grafik işlemci) aygıtı.
CL_DEVICE_DEFAULT	Sistem içindeki varsayılan OpenCL aygıtı. Varsayılan aygıt, CL_DEVICE_TYPE_CUSTOM aygıtı olamaz.

Tanımlayıcı	Açıklama
CL_DEVICE_CUSTOM	OpenCL C dilinde yazılmış programları desteklemeyen adanmış hızlandırıcılar.

Örnek:

```
void OnStart()
  {
   int cl_ctx;
//--- OpenCL bağlamını başlat
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
      Print("OpenCL bulunamadı");
      return;
     }
//--- OpenCL aygıtı ile ilgili genel bilgiyi görüntüle
   Print("OpenCL tipi: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
   Print("OpenCL tedarikçi kimliği: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("OpenCL birimleri: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("OpenCL frekansı: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY),"
   Print("OpenCL global bellek: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE),'
   Print("OpenCL yerel bellek: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE)," k
//--- free OpenCL context
   CLContextFree(cl_ctx);
  }
```

CLGetInfoString

OpenCL nesnesi veya aygıtı için istenen özelliğin string tipli değerine dönüş yapar.

```
bool  CLGetInfoString(
   int  handle,                          // OpenCL nesnesinin tanıtıcı değeri veya Op
   ENUM_OPENCL_PROPERTY_STRING  prop,    // istenen özellik
   string&  value                        // referans verilen dizgi
   );
```

Parametreler

handle

[in] OpenCL nesnesinin tanıtıcı değeri veya OpenCL aygıt numarası. OpenCL aygıtlarının numaralandırılmasına sıfır ile başlanır.

prop

[in] Alınması istenilen özellik değeri, ENUM_OPENCL_PROPERTY_STRING sayımının değerlerinden biri olabilir.

value

[out] Özellik değerinin alınacağı string tipli değişken.

Dönüş değeri

Başarılı ise 'true', değilse 'false'. Hata hakkında bilgi almak için GetLastError() fonksiyonunu kullanın.

ENUM_OPENCL_PROPERTY_STRING

Tanımlayıcı	Açıklama
CL_PLATFORM_PROFILE	CL_PLATFORM_PROFILE - OpenCL Profili. Profil ismi şu değerlerden biri olabilir: • FULL_PROFILE - uygulama OpenCL'i destekler (işlevsellik, OpenCL desteği için fazladan eklenti gerektirmeyen kernel özelliğinin bir parçası olarak tanımlanır); • EMBEDDED_PROFILE - uygulama OpenCL'i ek olarak destekler. Düzeltilmiş profil, her bir OpenCL sürümü için bir alt küme olarak tanımlanır.
CL_PLATFORM_VERSION	OpenCL versiyonu
CL_PLATFORM_VENDOR	Platform tedarikçisinin ismi
CL_PLATFORM_EXTENSIONS	Platform tarafından desteklenen eklentilerin listesi. Eklenti isimler, bu platforma bağlı tüm aygıtlar tarafından desteklenmelidir
CL_DEVICE_NAME	Aygıt ismi
CL_DEVICE_VENDOR	Tedarikçi ismi

Tanımlayıcı	Açıklama
CL_DRIVER_VERSION	major_number.minor_number biçiminde OpenCL sürücü versiyonu
CL_DEVICE_PROFILE	OpenCL aygıt profili. Profil ismi şu değerlerden biri olabilir: • FULL_PROFILE - uygulama OpenCL'i destekler (işlevsellik, OpenCL desteği için fazladan eklenti gerektirmeyen kernel özelliğinin bir parçası olarak tanımlanır); • EMBEDDED_PROFILE - uygulama OpenCL'i ek olarak destekler. Düzeltilmiş profil, her bir OpenCL sürümü için bir alt küme olarak tanımlanır.
CL_DEVICE_VERSION	"OpenCL<space><major_version.minor_version><space><vendor-specific information>" biçiminde OpenCL versiyonu
CL_DEVICE_EXTENSIONS	Aygıt tarafından desteklenen eklentilerin listesi. Bu liste, onaylanmış isimlerin yanında, tedarikçi tarafından sağlanan bir takım eklentileri de içerir: 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	Desteklenen kernellerin listesi";".
CL_DEVICE_OPENCL_C_VERSION	Bu aygıt için, derleyici tarafından desteklenen en yüksek versiyon. Versiyon biçimi: "OpenCL<space>C<space><major_version.minor_version><space><vendor-specific information> "
CL_ERROR_DESCRIPTION	OpenCL hatasının açıklaması

Örnek:

```
void OnStart()
  {
   int cl_ctx;
   string str;
//--- OpenCL bağlamını başlat
   if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
     {
      Print("OpenCL bulunamadı");
      return;
     }
```

```
//--- Platform ile ilgili bilgileri görüntüle
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_NAME,str))
      Print("OpenCL platform ismi: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VENDOR,str))
      Print("OpenCL platform tedarikçisi: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_VERSION,str))
      Print("OpenCL platform versiyonu: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_PROFILE,str))
      Print("OpenCL platform profili: ",str);
   if(CLGetInfoString(cl_ctx,CL_PLATFORM_EXTENSIONS,str))
      Print("OpenCL platform eklentileri: ",str);
//--- Aygıtla ilgili bilgileri görüntüle
   if(CLGetInfoString(cl_ctx,CL_DEVICE_NAME,str))
      Print("OpenCL aygıt ismi: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_PROFILE,str))
      Print("OpenCL aygıt profili: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_BUILT_IN_KERNELS,str))
      Print("OpenCL aygıtının kernelleri: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_EXTENSIONS,str))
      Print("OpenCL aygıt eklentileri: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VENDOR,str))
      Print("OpenCL aygıt tedarikçisi: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_VERSION,str))
      Print("OpenCL aygıt ver: ",str);
   if(CLGetInfoString(cl_ctx,CL_DEVICE_OPENCL_C_VERSION,str))
      Print("OpenCL open c versiyonu: ",str);
//--- OpenCL aygıtı ile ilgili genel bilgiyi görüntüle
   Print("OpenCL tipi: ",EnumToString((ENUM_CL_DEVICE_TYPE)CLGetInfoInteger(cl_ctx,CL_
   Print("OpenCL tedarikçi kimliği: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_VENDOR_ID));
   Print("OpenCL birimleri: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_COMPUTE_UNITS));
   Print("OpenCL frekansı: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_MAX_CLOCK_FREQUENCY));
   Print("OpenCL global bellek: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_GLOBAL_MEM_SIZE));
   Print("OpenCL yerel bellek: ",CLGetInfoInteger(cl_ctx,CL_DEVICE_LOCAL_MEM_SIZE));
//--- free OpenCL context
   CLContextFree(cl_ctx);
  }
```

CLContextCreate

Bir OpenCL bağlamı oluşturur ve onun tanıtıcı değerine dönüş yapar.

```
int  CLContextCreate(
   int  device=CL_USE_ANY     // OpenCL aygıtının seri numarası veya makro
   );
```

Parametre

device

[in] OpenCL aygıtının sistemdeki sıra numarası. Belirli bir değer yerine, aşağıdakilerden birini de kullanabilirsiniz:

- CL_USE_ANY - OpenCL sahip her aygıt desteklenir;
- CL_USE_CPU_ONLY - sadece CPU üzerindeki OpenCL emülasyonuna izin verilir;
- CL_USE_GPU_ONLY - OpenCL emülasyonuna izin verilmez ve sadece OpenCL desteğine sahip aygıtlar (ekran kartları) kullanılabilir;
- CL_USE_GPU_DOUBLE_ONLY - yalnızca double türünü destekleyen GPU'lara izin verilir.

Dönüş değeri

Başarı durumunda OpenCL bağlamının tanıtıcı değerine, aksi durumda -1 değerine dönüş yapar. Hata hakkında bilgi almak için GetLastError() fonksiyonunu kullanın.

CLContextFree

Bir OpenCL bağlamını siler.

```
void  CLContextFree(
   int  context     // OpenCL bağlamının tanıtıcı değeri
   );
```

Parametreler

context

[in] OpenCL bağlamının tanıtıcı değeri.

Dönüş değeri

None. İçsel hata durumunda _LastError değeri değişir. Hata hakkında daha fazla bilgi için GetLastError() fonksiyonunu kullanın.

CLGetDeviceInfo

OpenCL sürücüsünden aygıt özelliğini alır.

```
bool  CLGetDeviceInfo(
   int     handle,          // OpenCL aygıtının tanıtıcı değeri
   int     property_id,     // istenen özelliğin tanımlayıcısı
   uchar&  data[],          // verilerin alınacağı dizi
   uint&   size             // dizi elemanlarının sayısı, 0 varsayılan değerdir
   );
```

Parametreler

handle

[in] OpenCL aygıtının indisi veya CLContextCreate() fonksiyonu ile oluşturulmuş olan tanıtıcı değer.

property_id

[in] Alınması gereken OpenCL aygıtı özelliğinin kimliği (tanımlayıcısı). Bu değer, aşağıdaki tabloda belirtilen değerlerden biri olabilir.

data[]

[out] İstenen özellik değerlerinin alınması için bir dizi.

size

[out] *data[]* dizisinin büyüklük değeri.

Dönüş değeri

Başarılı ise 'true', değilse 'false'. Hata hakkında bilgi almak için GetLastError() fonksiyonunu kullanın.

Not

Tek boyutlu diziler için, OpenCL tamponuna yazılacak elemanların sayısı, AS_SERIES bayrağı hesaba katılarak hesaplanır.

Mevcut OpenCL aygıtı özelliklerinin listesi

Özelliğin tam açıklaması ve fonksiyonları OpenCL resmi internet sitesinde bulunabilir.

Tanımlayıcı	Değer
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
CL_DEVICE_PREFERRED_VECTOR_WIDTH_CHAR	0x1006

Tanımlayıcı	Değer
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
CL_DEVICE_MAX_CONSTANT_ARGS	0x1021
CL_DEVICE_LOCAL_MEM_TYPE	0x1022

Tanımlayıcı	Değer
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
CL_DEVICE_IMAGE_MAX_BUFFER_SIZE	0x1040
CL_DEVICE_IMAGE_MAX_ARRAY_SIZE	0x1041

Tanımlayıcı	Değer
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

Örnek:

```
void OnStart()
  {
//---
   int dCount= CLGetInfoInteger(0,CL_DEVICE_COUNT);
   for(int i = 0; i<dCount; i++)
     {
      int clCtx=CLContextCreate(i);
      if(clCtx == -1)
         Print("ERROR 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);
     }
  }
//--- Uzmanlar bültenindeki girdiler için bir örnek
//  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) i7-377
//  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

Bir kaynak kodundan OpenCL programı oluşturur.

```
int  CLProgramCreate(
   int           context,     // Bir OpenCL bağlamının tanıtıcı değeri
   const string  source       // Kaynak kodu
   );
```

OpenCL programı oluşturan ve derleyici mesajlarını referans ile geçirilen dizgi değişkene yazan aşırı yüklenmiş fonksiyon.

```
int  CLProgramCreate(
   int           context,     // Bir OpenCL bağlamının tanıtıcı değeri
   const string  source,      // Kaynak kodu
   string       &build_log    // Derleme kayıtlarını alması için eklenen dizgi değişke
   );
```

Parametreler

context

[in] OpenCL bağlamının tanıtıcı değeri.

source

[in] OpenCL programını içeren kaynak kodu dizesi.

&build_log

[in] OpenCL derleyicisinin mesajını almak için bir dizgi değişken.

Dönüş Değeri

Başarılı olması durumunda bir OpenCL nesnesinin tanıtıcı değeri. Hata durumunda -1 dönüşü yapar. Hata hakkında bilgi almak için GetLastError() fonksiyonunu kullanın.

Not

Hali hazırda kullanılmakta olan hata kodları şu şekildedir:

- ERR_OPENCL_INVALID_HANDLE - OpenCL bağlamı için hatalı tanıtıcı değer.
- ERR_INVALID_PARAMETER - hatalı string parametre.
- ERR_NOT_ENOUGH_MEMORY - işlemi tamamlamak için yeterli hafıza yok.
- ERR_OPENCL_PROGRAM_CREATE - OpenCL içsel hatası, veya derleme hatası.

Bazı ekran kartlarında varsayılan olarak double tipli sayılarla çalışılamaz. Bu durum 5105 numaralı hata değerine yol açabilir. double tipli sayılarla çalışabilmek için lütfen OpenCL programınıza şu direktifi ekleyin: #pragma OPENCL EXTENSION cl_khr_fp64 : enable. Ama ekran kartı double tipli sayıları desteklemiyorsa bu direktifin de bir yararı olmayacaktır.

Örnek:

```
//+------------------------------------------------------------------+
//| OpenCL kernel                                                    |
//+------------------------------------------------------------------+
const string
```

```
cl_src=
       //--- varsayılan olarak GPU double tipli sayıları desteklemiyor
       //--- double tipi ile çalışabilmek için cl_khr_fp64 direktifi kullanılır
       "#pragma OPENCL EXTENSION cl_khr_fp64 : enable      \r\n"
       //--- OpenCL kernel fonksiyonu
       "__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)
  {
//--- dizi büyüklüğünü denetle
   if(ArraySize(data)==0) return(false);
//--- dizi indisini denetle
   if(id>total_arrays) return(false);
//--- id numaralı dizi için yerel başlangıç payını hesapla
   int local_start_offset=id*N;
//--- elemanları 2 ile çarp
   for(int i=0; i<N; i++)
     {
      data[i+local_start_offset]*=2.0;
     }
   return true;
  }
//---
#define ARRAY_SIZE   100  // dizi büyüklüğü
#define TOTAL_ARRAYS 5    // dizi sayısı
//--- OpenCL işleyicileri
int cl_ctx;  // OpenCL bağlam işleyicisi
int cl_prg;  // OpenCL program işleyicisi
int cl_krn;  // OpenCL kernel işleyicisi
int cl_mem;  // OpenCL tampon işleyicisi
//---
double DataArray1[]; // CPU hesaplamaları için veri dizisi
double DataArray2[]; // GPU hesplamaları için veri dizisi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
```

```
int OnStart()
  {
//--- OpenCL nesnelerini başlat
//--- OpenCL bağlamı oluştur
   if((cl_ctx=CLContextCreate())==INVALID_HANDLE)
     {
      Print("OpenCL bulunamadı. Hata=",GetLastError());
      return(1);
     }
//--- OpenCL programı oluştur
   if((cl_prg=CLProgramCreate(cl_ctx,cl_src))==INVALID_HANDLE)
     {
      CLContextFree(cl_ctx);
      Print("OpenCL programı oluşturulamadı. Hata=",GetLastError());
      return(1);
     }
//--- OpenCL kerneli oluştur
   if((cl_krn=CLKernelCreate(cl_prg,"Test_GPU"))==INVALID_HANDLE)
     {
      CLProgramFree(cl_prg);
      CLContextFree(cl_ctx);
      Print("OpenCL kerneli oluşturulamadı. Hata=",GetLastError());
      return(1);
     }
//--- OpenCL tamponu oluştur
   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 tamponu oluşturulamadı. Hata=",GetLastError());
      return(1);
     }
//--- OpenCL kerneli içinsabit parametreleri ayarla
   CLSetKernelArgMem(cl_krn,0,cl_mem);
   CLSetKernelArg(cl_krn,1,ARRAY_SIZE);
   CLSetKernelArg(cl_krn,2,TOTAL_ARRAYS);
//--- veri dizilerini hazırla
   ArrayResize(DataArray1,ARRAY_SIZE*TOTAL_ARRAYS);
   ArrayResize(DataArray2,ARRAY_SIZE*TOTAL_ARRAYS);
//--- verileri dizilere gir
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- j numaralı dizi için yerel başlangıç payını hesapla
      uint local_offset=j*ARRAY_SIZE;
      //--- j numaralı diziyi hazırla
      for(int i=0; i<ARRAY_SIZE; i++)
        {
         //--- dizileri MathCos(i+j) fonksiyonu ile doldur;
```

```
         DataArray1[i+local_offset]=MathCos(i+j);
         DataArray2[i+local_offset]=MathCos(i+j);
        }
     };
//--- CPU hesaplamasını denetle
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- j numaralı dizinin hesaplanması
      Test_CPU(DataArray1,ARRAY_SIZE,j,TOTAL_ARRAYS);
     }
//--- CLExecute parametrelerini ayarla
   uint  offset[]={0};
//--- global iş büyüklüğü
   uint  work[]={TOTAL_ARRAYS};
//--- veriyi OpenCL tamponuna yaz
   CLBufferWrite(cl_mem,DataArray2);
//--- OpenCL kernelini çalıştır
   CLExecute(cl_krn,1,offset,work);
//--- OpenCL tamponundan verileri oku
   CLBufferRead(cl_mem,DataArray2);
//--- toplam hata
   double total_error=0;
//--- sonuçları karşılaştır ve htayı hesapla
   for(int j=0; j<TOTAL_ARRAYS; j++)
     {
      //--- j numaralı dizi için yerel başlangıç payını hesapla
      uint local_offset=j*ARRAY_SIZE;
      //--- sonuçları karşılaştır
      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;
         //--- ilk ve son dizileri göster
         if((j==0) || (j==TOTAL_ARRAYS-1))
            PrintFormat("%d dizi arasından %d numarlı dizi, eleman [%d]:  %f, %f, [hat
        }
     }
   PrintFormat("Toplam hata: %f",total_error);
//--- OpenCL nesnelerini sil
//--- OpenCL tamponunu boşlalt
   CLBufferFree(cl_mem);
//--- OpenCL kernelini serbest bırak
   CLKernelFree(cl_krn);
//--- OpenCL programını srbest bırak
   CLProgramFree(cl_prg);
//--- OpenCL bağlamıı serbest bırak
   CLContextFree(cl_ctx);
```

```
//---
   return(0);
  }
```

CLProgramFree

Bir OpenCL programını siler.

```
void  CLProgramFree(
   int  program     // Bir OpenCL nesnesinin tanıtıcı değeri
   );
```

Parametreler

program

[in] OpenCL nesnesinin tanıtıcı değeri.

Dönüş değeri

None. İçsel hata durumunda _LastError değeri değişir. Hata hakkında daha fazla bilgi için GetLastError() fonksiyonunu kullanın.

CLKernelCreate

OpenCL programının kernelini oluşturur ve onun tanıtıcı değerine dönüş yapar.

```
int  CLKernelCreate(
   int           program,        // Bir OpenCL nesnesinin tanıtıcı değeri
   const string  kernel_name     // Kernel ismi
   );
```

Parametreler

program

[in] OpenCL programının bir nesnesinin tanıtıcı değeri.

kernel_name

[in] Çalışmanın başlayacağı uygun OpenCL programındaki kernel fonksiyonunun ismi.

Dönüş değeri

Başarılı olması durumunda bir OpenCL nesnesinin tanıtıcı değeri. Hata durumunda -1 dönüşü yapar. Hata hakkında bilgi almak için GetLastError() fonksiyonunu kullanın.

Not

Hali hazırda kullanılmakta olan hata kodları şu şekildedir:

- ERR_OPENCL_INVALID_HANDLE - OpenCL *programı* için hatalı tanıtıcı değer.
- ERR_INVALID_PARAMETER - hatalı parametre.
- ERR_OPENCL_TOO_LONG_KERNEL_NAME - kernel ismi 127 karakterden fazla içeriyor.
- ERR_OPENCL_KERNEL_CREATE - OpenCL nesnesi oluşturulurken bir içsel hata ile karşılaşıldı.

CLKernelFree

Bir OpenCL başlatma fonksiyonunu siler.

```
void  CLKernelFree(
   int  kernel     // Bir OpenCL programının kernelinin tanıtıcı değeri
   );
```

Parametreler

kernel_name

[in] Kernel nesnesinin tanıtıcı değeri.

Dönüş değeri

None. İçsel hata durumunda _LastError değeri değişir. Hata hakkında daha fazla bilgi için GetLastError() fonksiyonunu kullanın.

CLSetKernelArg

OpenCL fonksiyonu için bir parametre ayarlar.

```
bool  CLSetKernelArg(
   int   kernel,        // Bir OpenCL programının kernelinin tanıtıcı değeri
   uint  arg_index,     // OpenCL fonksiyonu için argüman numarası
   void  arg_value      // Kaynak kodu
   );
```

Parametreler

kernel

[in] Bir OpenCL programının kernelinin tanıtıcı değeri.

arg_index

[in] Fonksiyon argümanının numarası - numaralama sıfır ile başlar.

arg_value

[in] Fonksiyon argümanının değeri.

Dönüş değeri

Başarı durumunda 'true', aksi durumda 'false' dönüşü yapar. Hata hakkında bilgi almak için GetLastError() fonksiyonunu kullanın.

Not

Hali hazırda kullanılmakta olan hata kodları şu şekildedir:

- ERR_INVALID_PARAMETER,
- ERR_OPENCL_INVALID_HANDLE - OpenCL kerneli için hatalı tanıtıcı değer.
- ERR_OPENCL_SET_KERNEL_PARAMETER - İçsel OpenCL hatası.

CLSetKernelArgMem

Bir OpenCL tamponunu, OpenCL fonksiyonunun parametresi olarak ayarlar.

```
bool  CLSetKernelArgMem(
   int   kernel,           // Bir OpenCL programının kernelinin tanıtıcı değeri
   uint  arg_index,        // OpenCL fonksiyonu için argüman numarası
   int   cl_mem_handle     // OpenCL tamponunun tanıtıcı değeri
   );
```

Parametreler

kernel

[in] Bir OpenCL programının kernelinin tanıtıcı değeri.

arg_index

[in] Fonksiyon argümanının numarası - numaralama sıfır ile başlar.

cl_mem_handle

[in] Bir OpenCL tamponunun tanıtıcı değeri.

Dönüş değeri

Başarı durumunda 'true', aksi durumda 'false' dönüşü yapar. Hata hakkında bilgi almak için GetLastError() fonksiyonunu kullanın.

CLSetKernelArgMemLocal

Yerel tamponu kernel fonksiyonunun bir argümanı olarak ayarlar.

```
bool  CLSetKernelArgMemLocal(
   int    kernel,           // OpenCL programı kernelinin işleyicisi
   uint   arg_index,        // OpenCL fonksiyonunun argüman numarası
   ulong  local_mem_size    // tampon boyutu
   );
```

Parametreler

kernel

[in] Bir OpenCL programının kernelinin tanıtıcı değeri.

arg_index

[in] Fonksiyon argümanının numarası - numaralama sıfır ile başlar.

local_mem_size

[in] Tampon büyüklüğü.

Dönüş Değeri

Başarı durumunda 'true', aksi durumda 'false' dönüşü yapar. Hata hakkında bilgi almak için GetLastError() fonksiyonunu kullanın.

CLBufferCreate

Bir OpenCL tamponu oluşturur ve onun tanıtıcı değerine dönüş yapar.

```
int  CLBufferCreate(
   int   context,     // Bir OpenCL bağlamının tanıtıcı değeri
   uint  size,        // Tampon büyüklüğü
   uint  flags        // OpenCL tamponunun özelliklerini belirleyen bayrak kombinasyon
   );
```

Parametreler

context

[in] Bir OpenCL bağlamının tanıtıcı değeri.

size

[in] Tampon büyüklüğü.

bayraklar

[in] Bayrakların kombinasyonu ile ayarlanan tampon özellikleri: CL_MEM_READ_WRITE, CL_MEM_WRITE_ONLY, CL_MEM_READ_ONLY, CL_MEM_ALLOC_HOST_PTR.

Dönüş değeri

Eğer sonuç başarılıysa OpenCL tamponunun tanıtıcı değeri. Hata durumunda -1 dönüşü yapar. Hata hakkında bilgi almak için GetLastError() fonksiyonunu kullanın.

Not

Hali hazırda kullanılmakta olan hata kodları şu şekildedir:

- ERR_OPENCL_INVALID_HANDLE - OpenCL bağlamı için hatalı tanıtıcı değer.
- ERR_NOT_ENOUGH_MEMORY - yetersiz bellek.
- ERR_OPENCL_BUFFER_CREATE - tamponların oluşturulması sırasında içsel hata.

CLBufferFree

Bir OpenCL tamponunu siler.

```
void  CLBufferFree(
   int   buffer     // OpenCL tamponunun tanıtıcı değeri
   );
```

Parametreler

buffer

[in] Bir OpenCL tamponunun tanıtıcı değeri.

Dönüş değeri

None. İçsel hata durumunda _LastError değeri değişir. Hata hakkında daha fazla bilgi için GetLastError() fonksiyonunu kullanın.

CLBufferWrite

Bir diziden OpenCL tamponuna veri yazar ve yazılan elemanların sayısına dönüş yapar.

```
uint  CLBufferWrite(
   int           buffer,                    // OpenCL tamponunun tanıtıcı değeri
   const void&   data[],                    // değerler için dizi
   uint          buffer_offset=0,           // OpenCL tamponunda bayt bazında konum değ
   uint          data_offset=0,             // Eleman sayısı bazında konum değeri, vars
   uint          data_count=WHOLE_ARRAY     // Dizideki yazılacak verilerin sayısı, va
   );
```

Matrisleri ve vektörleri işlemek için versiyonlar da mevcuttur.

Matristen değerleri arabelleğe yazar ve başarılı olursa true değerini geri döndürür.

```
uint  CLBufferWrite(
   int           buffer,                     // OpenCL arabelleğine olan tanıtıcı
   uint          buffer_offset,              // OpenCL arabelleğinde bayt cinsinden ofs
   matrix<T>     &mat                        // arabelleğe yazılacak değerleri içeren m
   );
```

Vektörden değerleri arabelleğe yazar ve başarılı olursa true değerini geri döndürür.

```
uint  CLBufferWrite(
   int           buffer,                     // OpenCL arabelleğine olan tanıtıcı
   uint          buffer_offset,              // OpenCL arabelleğinde bayt cinsinden ofs
   vector<T>     &vec                       // arabelleğe yazılacak değerleri içeren ve
   );
```

Parametreler

buffer

[in] Bir OpenCL tamponunun tanıtıcı değeri.

data[]

[in] OpenCL tamponuna yazılacak verilerden oluşan dizi. Referans ile geçirilir.

buffer_offset

[in] OpenCL tamponu içinde bayt bazında yazma başlangıç konumu. Yazma işlemi, varsayılan olarak tamponun en başından başlar.

data_offset

[in] OpenCL tamponuna değerleri yazmak için, ilk dizi elemanının indisi. Varsayılan olarak dizinin en başındaki değerler alınır.

data_count

[in] Yazılması gereken değerlerin sayısı. Varsayılan olarak, tüm dizi elemanları.

&mat

[out] Verileri arabelleğe yazılacak matris şu üç türden herhangi biri olabilir: matrix, matrixf veya matrixc.

&vec

[out] Verileri arabelleğe yazılacak vektör şu üç türden herhangi biri olabilir: vector, vectorf veya vectorc.

Dönüş değeri

Yazılan elemanların sayısı. Hata durumunda 0 dönüşü yapılır. Hata hakkında bilgi almak için GetLastError() fonksiyonunu kullanın.

Matris veya vektör başarıyla işlenirse true, aksi takdirde false geri döndürür.

Not

Tek boyutlu diziler için, OpenCL tamponuna yazılacak elemanların sayısı, AS_SERIES bayrağı hesaba katılarak hesaplanır.

İki veya daha fazla boyuta sahip diziler, tek boyutlu farz edilir. Bu durumda, *data_offset* parametresi, ilk boyuttaki elemanların sayısı değil, sunulan dizide atlanması gereken elemanların sayısıdır.

OpenCL'de MatMul metodu ve paralel hesaplama kullanılarak matris çarpımına **örnek**:

```
#define M       3000      // birinci matristeki satır sayısı
#define K       2000      // birinci matristeki sütun sayısı ikinci matristeki satır s
#define N       3000      // ikinci matristeki sütun sayısı

//+------------------------------------------------------------------+
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";
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
void OnStart()
 {
//--- rastgele sayı üreticisini başlat
  MathSrand((int)TimeCurrent());
//--- matrisleri rastgele değerlerle doldur
```

```
  matrixf mat1(M, K, MatrixRandom) ;     // birinci matris
  matrixf mat2(K, N, MatrixRandom);      // ikinci matris

//--- saf yöntem kullanarak matrislerin çarpımını hesapla
  uint start=GetTickCount();
  matrixf matrix_naive=matrixf::Zeros(M, N);// iki matrisin çarpılmasının sonucu burac
  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;

//--- MatMul aracılığıyla matrislerin çarpımını hesapla
  start=GetTickCount();
  matrixf matrix_matmul=mat1.MatMul(mat2);
  uint time_matmul=GetTickCount()-start;

//--- OpenCL'de matrislerin çarpımını hesapla
  matrixf matrix_opencl=matrixf::Zeros(M, N);
  int cl_ctx;              // içerik tanıtıcısı
  if((cl_ctx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
   {
    Print("OpenCL bulunamadı, çıkış yapılıyor");
    return;
   }
  int cl_prg;              // program tanıtıcısı
  int cl_krn;              // çekirdek tanıtıcısı
  int cl_mem_in1;          // birinci (girdi) arabellek tanıtıcısı
  int cl_mem_in2;          // ikinci (girdi) arabellek tanıtıcısı
  int cl_mem_out;          // üçüncü (çıktı) arabellek tanıtıcısı
//--- programı ve çekirdeği oluştur
  cl_prg = CLProgramCreate(cl_ctx, clSrc);
  cl_krn = CLKernelCreate(cl_prg, "matricesMul");
//--- üç matris için üç arabelleğin tümünü oluştur
  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);
//--- üçüncü matris - çıktı
  cl_mem_out=CLBufferCreate(cl_ctx, M*N*sizeof(float), CL_MEM_READ_WRITE);
//--- çekirdek argümanlarını ayarla
  CLSetKernelArgMem(cl_krn, 0, cl_mem_in1);
  CLSetKernelArgMem(cl_krn, 1, cl_mem_in2);
  CLSetKernelArgMem(cl_krn, 2, cl_mem_out);
//--- matrisleri cihaz arabelleklerine yaz
  CLBufferWrite(cl_mem_in1, 0, mat1);
  CLBufferWrite(cl_mem_in2, 0, mat2);
  CLBufferWrite(cl_mem_out, 0, matrix_opencl);
//--- OpenCL kodunu çalıştırma zamanı başlangıcı
  start=GetTickCount();
//--- görev çalışma alanının parametrelerini ayarla ve OpenCL programını çalıştır
```

```
  uint  offs[2] = {0, 0};
  uint works[2] = {M, N};
  start=GetTickCount();
  bool ex=CLExecute(cl_krn, 2, offs, works);
//--- sonucu hesapla
  if(CLBufferRead(cl_mem_out, 0, matrix_opencl))
    PrintFormat("[%d x %d] matris okuması: ", matrix_opencl.Rows(), matrix_opencl.Cols
   else
      Print("CLBufferRead(cl_mem_out, 0, matrix_opencl başarısız oldu. Hata ",GetLastE
  uint time_opencl=GetTickCount()-start;
  Print("Her yöntemi kullanarak hesaplama sürelerini karşılaştır");
  PrintFormat("Saf yöntemle çarpma süresi = %d ms",time_naive);
  PrintFormat("MatMul metoduyla çarpma süresi = %d ms",time_matmul);
  PrintFormat("OpenCl ile çarpma süresi = %d ms",time_opencl);
//--- tüm OpenCL içeriklerini bırak
  CLFreeAll(cl_ctx, cl_prg, cl_krn, cl_mem_in1, cl_mem_in2, cl_mem_out);

//--- tüm sonuç matrisleri birbirleriyle karşılaştır
  Print("Sonuç matrisler arasında kaç tane tutarsızlık hatası var?");
  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);
/*
  Sonuç:

   [3000 x 3000] matris okuması:
  Her yöntemi kullanarak hesaplama sürelerini karşılaştır
   Saf yöntemle çarpma süresi = 54750 ms
   MatMul metoduyla çarpma süresi = 4578 ms
   OpenCl ile çarpma süresi = 922 ms
  Sonuç matrisler arasında kaç tane tutarsızlık hatası var?
   matrix_direct.Compare(matrix_matmul,1e-12)=0
   matrix_matmul.Compare(matrix_opencl,1e-12)=0
*/
 }
//+------------------------------------------------------------------+
//| Matrisi rastgele değerlerle doldur                               |
//+------------------------------------------------------------------+
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.);
     }
   }
 }
```

```
//+------------------------------------------------------------------+
//| Tüm OpenCL içeriklerini bırak                                    |
//+------------------------------------------------------------------+
void CLFreeAll(int cl_ctx, int cl_prg, int cl_krn,
               int cl_mem_in1, int cl_mem_in2, int cl_mem_out)
 {
//--- OpenCL tarafından ters sırada oluşturulan tüm içerikleri sil
  CLBufferFree(cl_mem_in1);
  CLBufferFree(cl_mem_in2);
  CLBufferFree(cl_mem_out);
  CLKernelFree(cl_krn);
  CLProgramFree(cl_prg);
  CLContextFree(cl_ctx);
 }
```

CLBufferRead

Bir OpenCL tamponunu bir diziye okur ve okunan eleman sayısına dönüş yapar.

```
uint  CLBufferRead(
   int           buffer,                  // OpenCL tamponunun tanıtıcı değeri
   const void&   data[],                  // değerler için dizi
   uint          buffer_offset=0,         // OpenCL tamponunda bayt bazında konum değ
   uint          data_offset=0,           // Eleman sayısı bazında konum değeri, vars
   uint          data_count=WHOLE_ARRAY   // Tampondaki okunacak verilerin sayısı, va
   );
```

Matrisleri ve vektörleri işlemek için versiyonlar da mevcuttur.

OpenCL arabelleğini matrise okur ve başarılı olursa true değerini geri döndürür.

```
uint  CLBufferRead(
   int           buffer,                  // OpenCL arabelleğine olan tanıtıcı
   uint          buffer_offset,           // OpenCL arabelleğinde bayt cinsinden ofs
   const matrxi& mat,                     // arabellekten değerler alacak matris
   ulong         rows=-1,                 // matristeki satır sayısı
   ulong         cols=-1                  // matristeki sütun sayısı
   );
```

OpenCL arabelleğini vektöre okur ve başarılı olursa true değerini geri döndürür.

```
uint  CLBufferRead(
   int           buffer,                  // OpenCL arabelleğine olan tanıtıcı
   uint          buffer_offset,           // OpenCL arabelleğinde bayt cinsinden ofs
   const vector& vec,                     // arabellekten değerler alacak vektör
   ulong         size-1,                  // vektör uzunluğu
   );
```

Parametreler

buffer

[in] OpenCL tamponu için bir tanıtıcı değer.

data[]

[in] OpenCL tamponunun değerlerini almak için bir dizi. Referans ile geçirilir.

buffer_offset

[in] OpenCL tamponunda okuma işleminin başlayacağı bayt bazındaki konum değeri. Okuma, varsayılan olarak tamponun en başından başlar.

data_offset

[in] OpenCL tamponunun değerlerini yazmak için, ilk dizi elemanının indisi. Okunan elemanların yazımına, varsayılan olarak sıfırdan başlanır.

data_count

[in] Okunması gereken değerlerin sayısı. Varsayılan olarak OpenCL tamponunun tamamı okunur.

mat

[out] Arabellekten veriler alacak matris şu üç türden herhangi biri olabilir: matrix, matrixf veya matrixc.

vec

[out] Arabellekten veriler alacak vektör şu üç türden herhangi biri olabilir: vector, vectorf veya vectorc.

rows=-1

[in] Bu parametre belirtilirse, *cols* parametresi de belirtilmelidir. Yeni matris boyutları belirtilmezse, mevcut olanlar kullanılır. Değer -1 olarak belirtilirse satır sayısı değişmez.

cols=-1

[in] Bu parametre belirtilmezse, *rows* parametresi de belirtilmemelidir. Matris şu kurala bağlıdır: ya her iki parametre de belirtilir ya da hiçbiri belirtilmez, aksi takdirde hata meydana gelir. Her iki parametre de (*rows* ve *cols*) belirtilirse, matris boyutu değiştirilir. Değer -1 olarak belirtilirse sütun sayısı değişmez.

size=-1

[in] Bu parametre belirtilmezse veya -1 olarak belirtilirse vektör uzunluğu değişmez.

Dönüş değeri

Okunan elemanların sayısı. Hata durumunda 0 dönüşü yapılır. Hata hakkında bilgi almak için GetLastError() fonksiyonunu kullanın.

Matris veya vektör başarıyla işlenirse true, aksi takdirde false geri döndürür.

Not

Tek boyutlu diziler için, OpenCL tamponuna yazılacak elemanların sayısı, AS_SERIES bayrağı hesaba katılarak hesaplanır.

İki veya daha fazla boyuta sahip diziler, tek boyutlu farz edilir. Bu durumda, *data_offset* parametresi, ilk boyuttaki elemanların sayısı değil, sunulan dizide atlanması gereken elemanların sayısıdır.

Denklem kullanılarak pi sayısının hesaplamasına **örnek**:

$$\pi = \lim_{N\to\infty}\frac{1}{N}\sum_{k=0}^{N-1}\frac{4}{1+\left(\frac{2k+1}{2N}\right)^2} = 16\lim_{N\to\infty}N\sum_{k=0}^{N-1}\frac{1}{4N^2+(2k+1)^2}$$

```
#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); }

//--- OpenCL programı kodu
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";

//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
int OnStart()
 {
  Print("Pi sayısının hesaplanması: adım = "+D2S(_step, 12)+"; _intrnCnt = "+I2S(_intr
//--- OpenCL içeriklerini hazırla
  int clCtx;
  if((clCtx=CLContextCreate(CL_USE_GPU_ONLY))==INVALID_HANDLE)
   {
    Print("OpenCL bulunamadı");
    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};
//--- OpenCL programını başlat
  ulong start=GetMicrosecondCount();
  if(!CLExecute(clKrn, 1, offs, works))
   {
```

```
     Print("CLExecute(clKrn, 1, offs, works) başarısız oldu! Hata ",GetLastError());
     CLFreeAll(clMem, clKrn, clPrg, clCtx);
     return(-1);
    }
//--- OpenCL cihazından sonuçları al
  vector buffer(_divisor);
  if(!CLBufferRead(clMem, 0, buffer))
    {
     Print("CLBufferRead(clMem, 0, buffer) başarısız oldu! Hata ",GetLastError());
     CLFreeAll(clMem, clKrn, clPrg, clCtx);
     return(-1);
    }
//--- tüm değerleri pi sayısını hesaplamak için topla
  double Pi=buffer.Sum()*_step;

  double time=(GetMicrosecondCount()-start)/1000.;
  Print("OpenCL: pi sayısı hesaplandı, geçen süre:"+D2S(time, 2)+" ms");
  Print("Pi sayısı = "+DoubleToString(Pi, 12));
//--- hafızayı temizle
  CLFreeAll(clMem, clKrn, clPrg, clCtx);
//--- başarılı
  return(0);
 }
  /*
  Pi sayısının hesaplanması: adım = 0.000000001000; _intrnCnt = 25000
  OpenCL: GPU device 'Ellesmere' selected
  OpenCL: pi sayısı hesaplandı, geçen süre: 99.98 ms
  Pi sayısı = 3.141592653590
  */
//+------------------------------------------------------------------+
//| Hafızayı temizlemek için yardımcı kod parçası                    |
//+------------------------------------------------------------------+
void CLFreeAll(const int clMem, const int clKrn, const int clPrg, const int clCtx)
 {
  CLBufferFree(clMem);
  CLKernelFree(clKrn);
  CLProgramFree(clPrg);
  CLContextFree(clCtx);
 }
```

CLExecute

Bir OpenCL programını çalıştırır. Fonksiyonun 3 versiyonu bulunmaktadır:

1. Tek bir kernel kullanarak, Kernel fonksiyonlarını çalıştırır

```
bool  CLExecute(
   int           kernel                    // Bir OpenCL programının kernelinin tanıtıc
   );
```

2. Görev alanı açıklaması ile birkaç kernel (OpenCL fonksiyonu) kopyasını çalıştırır

```
bool  CLExecute(
   int           kernel,                   // Bir OpenCL programının kernelinin tanıtıc
   uint          work_dim,                 // Görev alanının boyutu
   const uint&   global_work_offset[],     // görev alanındaki başlangıç konumu
   const uint&   global_work_size[]        // Toplam görev sayısı
   );
```

3. Görev alanı açıklaması ile, grubun yerel görevler alt-kümesinin büyüklüğünü belirterek, birkaç kernel (OpenCL fonksiyonu) kopyasını çalıştırır

```
bool  CLExecute(
   int           kernel,                   // Bir OpenCL programının kernelinin tanıtıc
   uint          work_dim,                 // Görev alanının boyutu
   const uint&   global_work_offset[],     // görev alanındaki başlangıç konumu
   const uint&   global_work_size[],       // Toplam görev sayısı
   const uint&   local_work_size[]         // Yerel gruptaki görevlerin sayısı
   );
```

Parametreler

kernel

[in] OpenCL kernelinin tanıtıcı değeri.

work_dim

[in] Görev alanının boyutu.

global_work_offset[]

[in] Görev alanındaki başlangıç konumu.

global_work_size[]

[in] Görevler alt-kümesinin büyüklüğü.

local_work_size[]

[in] Grubun yerel görevler alt-kümesinin büyüklüğü.

Dönüş değeri

Başarı durumunda 'true', aksi durumda 'false' dönüşü yapar. Hata hakkında bilgi almak için GetLastError() fonksiyonunu kullanın.

Not

Aşağıdaki örnekte yer alan parametre kullanımını göz önüne alın:

- *work_dim*, görevi tanımlayan *work_items[]* dizisini belirtir. *work_dim*=3 ise, üç boyutlu dizi kullanılır *work_items[N1, N2, N3]* is used.
- *global_work_size[]*, *work_items[]* dizisinin büyüklüğünü ayarlayan değerler içerir. work_dim=3 ise, *global_work_size[3]* dizisi {40, 100, 320} olabilir. Böylece *work_items[40, 100, 320]* dizisini elde ederiz. Yani görevlerin toplam sayısı: 40 x 100 x 320 = 1 280 000.
- *local_work_size[]* OpenCL programının belirtilen kerneli tarafından çalıştırılacak görevlerin alt-kümesini ayarlar. Bunun boyutu *work_items[]* dizisinin boyutuna eşittir ve görevler alt kümesinin daha küçük alt kümelere bçlünmesine izin verir. *work_items[]* global görevler kümesinin daha küçük alt-kümelere bölünebilmesi için *local_work_size[]* dizisi seçilmelidir. *local_work_size[3] ={10, 10, 10}* mevcut örneğe uyacaktır; *work_items[40, 100, 320]* dizisi, herhangi bir aşım olmadan *local_items[10, 10, 10]* dizisinden toplanabilir.

CLExecutionStatus

OpenCL programının çalışma durumuna dönüş yapar.

```
int  CLExecutionStatus(
   int   kernel            // OpenCL programının işleyicisi
   );
```

Parametreler

kernel

[in] Bir OpenCL programının kernelinin tanıtıcı değeri.

Dönüş Değeri

OpenCL programının çalışma durumuna dönüş yapar. Dönüş değeri şunlardan biri olabilir:

- CL_COMPLETE=0 - program tamamlandı,
- CL_RUNNING=1 - çalışıyor,
- CL_SUBMITTED=2 - çalıştırma için sunuldu,
- CL_QUEUED=3 - sıraya eklendi,
- -1 (eksi bir) - CLExecutionStatus() çalıştırılırken bir hata oluştu.

Veritabanlarıyla çalışma

Veritabanlarıyla çalışma fonksiyonları popüler ve kullanımı kolay SQLite motorunu kullanır. Bu motorun önemli bir özelliği, tüm veritabanının kullanıcının bilgisayarında bulunan tek bir standart dosyaya yerleştirilmiş olmasıdır.

Fonksiyonlar, basit SQL istekleri kullanılarak tabloların kolayca oluşturulmasına, bunlara veri eklenmesine, değişiklik ve örnekleme yapılmasına olanak tanır:

- herhangi bir formattan işlem geçmişi ve kotasyon alma,
- optimizasyon ve sınama sonuçlarını kaydetme,
- diğer analiz paketleriyle veri hazırlama ve veri değişimi,
- MQL5 uygulama ayarlarını ve durumunu saklama.

Sorgular, istatistiksel ve matematiksel fonksiyonların kullanılmasına olanak sağlar.

Veritabanlarıyla çalışma fonksiyonları, büyük veri dizilerini işlemek için en çok yinelenen işlemlerin SQL istekleriyle değiştirilmesine olanak tanır; bu, birçok durumda karmaşık döngüler ve karşılaştırmaları programlamak yerine DatabaseExecute/DatabasePrepare çağrılarını kullanmanıza olanak sağlar. İstek sonuçlarını hazır bir yapı halinde kolayca elde etmek için DatabaseReadBind fonksiyonunu kullanın. Fonksiyon, tek bir çağrıda tüm kayıt alanlarının aynı anda okunmasına olanak sağlar.

Okuma, yazma ve değiştirme işlemlerini hızlandırmak için, veritabanı DATABASE_OPEN_MEMORY bayrağıyla RAM'de açılabilir/oluşturulabilir, ancak bu tür bir veritabanı yalnızca belirli bir uygulama tarafından kullanılabilir ve paylaşılmaz. Sabit diskte bulunan veritabanlarıyla çalışırken, toplu veri eklemeleri/değişiklikleri DatabaseTransactionBegin/DatabaseTransactionCommit/DatabaseTransactionRollback kullanılarak sarılmalıdır. Bu, işlemi yüzlerce kez hızlandırır.

Fonksiyonlarla çalışmaya başlamak için SQLite: MQL5'te SQL veritabanlarıyla yerel çalışma makalesini okuyun.

Fonksiyon	Eylem
DatabaseOpen	Belirtilen dosyada veritabanı açar veya oluşturur
DatabaseClose	Veritabanını kapatır
DatabaseImport	Verileri bir dosyadan tabloya aktarır
DatabaseExport	Bir tablo veya SQL isteği yürütme sonucunu bir CSV dosyasına aktarır
DatabasePrint	Uzman Danışmanlar günlüğüne bir tablo veya SQL isteği yürütme sonucu yazdırır
DatabaseTableExists	Veritabanında tablonun varlığını kontrol eder
DatabaseExecute	Belirtilen veritabanına istek yürütür
DatabasePrepare	Daha sonra DatabaseRead() kullanılarak yürütülebilen bir istek tanıtıcı değeri oluşturur
DatabaseReset	DatabasePrepare() çağrıldıktan sonra olduğu gibi isteği sıfırlar

Fonksiyon	Eylem
DatabaseBind	İstekte bir parametre değeri ayarlar
DatabaseBindArray	Diziyi parametre değeri olarak ayarlar
DatabaseRead	İstek sonucunda bir sonraki girdiye gider
DatabaseFinalize	DatabasePrepare()'da oluşturulan isteği kaldırır
DatabaseTransactionBegin	İşlem yürütmeyi başlatır
DatabaseTransactionCommit	İşlem yürütmeyi tamamlar
DatabaseTransactionRollback	İşlemleri geri alır
DatabaseColumnsCount	İstekteki alan sayısını elde eder
DatabaseColumnName	Alan adını indekse göre elde eder
DatabaseColumnType	Alan tipini indekse göre elde eder
DatabaseColumnSize	Alan boyutunu bayt cinsinden elde eder
DatabaseColumnText	Alan değerini geçerli kayıttan dizge olarak elde eder
DatabaseColumnInteger	Geçerli kayıttan int tipi değeri elde eder
DatabaseColumnLong	Geçerli kayıttan long tipi değeri elde eder
DatabaseColumnDouble	Geçerli kayıttan double tipi değeri elde eder
DatabaseColumnBlob	Alan değerini geçerli kayıttan dizi olarak elde eder

İstatistiksel fonksiyonlar:

- mode - mod
- median - medyan (50'nci persantil)
- percentile_25 - 25'inci persantil
- percentile_75
- percentile_90
- percentile_95
- percentile_99
- stddev or stddev_samp — örneklem standart sapması
- stddev_pop — popülasyon standart sapması
- variance or var_samp — örneklem varyansı
- var_pop — popülasyon varyansı

Matematiksel fonksiyonlar:

- acos(X) - radyan cinsinden arccos
- acosh(X) - hiperbolik arccos
- asin(X) - radyan cinsinden arcsin

- asinh(X) - hiperbolik arcsin
- atan(X) - radyan cinsinden arctan
- atan2(X,Y) - X/Y oranlı radyan cinsinden arctan
- atanh(X) - hiperbolik arctan
- ceil(X) - yukarıdaki en yakın tam sayıya yuvarlar
- ceiling(X) - yukarıdaki en yakın tam sayıya yuvarlar
- cos(X) - radyan cinsinden açının kosinüsü
- cosh(X) - hiperbolik kosinüs
- degrees(X) - radyanı açıya dönüştürür
- exp(X) - üs
- floor(X) - aşağıdaki en yakın tam sayıya yuvarlar
- ln(X) - doğal logaritma
- log(B,X) - belirtilen tabana göre logaritma
- log(X) - bayağı logaritma
- log10(X) - bayağı logaritma
- log2(X) - 2 tabanına göre logaritma
- mod(X,Y) - bölme işleminin kalanı
- pi() - yaklaşık Pi
- pow(X,Y) - belirtilen tabanı kuvvet yapar
- power(X,Y) - belirtilen tabanı kuvvet yapar
- radians(X) - açıyı radyana dönüştürür
- sin(X) - radyan cinsinden açının sinüsü
- sinh(X) - hiperbolik sinüs
- sqrt(X) - karekök
- tan(X) - radyan cinsinden açının tanjantı
- tanh(X) - hiperbolik tanjant
- trunc(X) - 0'a yakın olan tam sayıya kısaltır

Örnek:

```
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

Belirtilen dosyada veritabanı açar veya oluşturur.

```
int  DatabaseOpen(
   string  filename,       // dosya adı
   uint    flags           // bayrak kombinasyonu
   );
```

Parametreler

filename

[in] "MQL5\Files" dizinindeki dosya adı.

flags

[in] ENUM_DATABASE_OPEN_FLAGS listesinden bayrak kombinasyonu.

Geri dönüş değeri

Başarılı bir şekilde yürütülürse, fonksiyon veritabanına erişmek için kullanılan veritabanı tanıtıcı değerini geri döndürür. Aksi takdirde, INVALID_HANDLE geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_INTERNAL_ERROR (4001) - kritik çalışma zamanı hatası;
- ERR_WRONG_INTERNAL_PARAMETER (4002) - "MQL5\Files" klasörüne erişirken dahili hata oluştu;
- ERR_INVALID_PARAMETER (4003) - veritabanı dosyasının yolu boş bir dizge içeriyor ya da uyumsuz bir bayrak kombinasyonu ayarlanmış;
- ERR_NOT_ENOUGH_MEMORY (4004) - yetersiz bellek;
- ERR_WRONG_FILENAME (5002) - yanlış veritabanı dosyası adı;
- ERR_TOO_LONG_FILENAME (5003) - veritabanı dosyasının mutlak yolu maksimum uzunluğu aşıyor;
- ERR_DATABASE_TOO_MANY_OBJECTS (5122) - kabul edilebilir maksimum Database nesnesi sayısı aşıldı;
- ERR_DATABASE_CONNECT (5123) - veritabanı bağlantı hatası;
- ERR_DATABASE_MISUSE (5621) - incorrect use of the SQLite library.

Not

filename parametresi NULL ya da boş dizge "" içeriyorsa, diskte geçici bir dosya oluşturulur. Bu geçici dosya, veritabanı bağlantısı kapatıldıktan sonra otomatik olarak silinir.

filename parametresi ":memory:" içeriyorsa, veritabanı bellekte oluşturulur ve bağlantı kapatıldıktan sonra otomatik olarak silinir.

flags parametresi DATABASE_OPEN_READONLY veya DATABASE_OPEN_READWRITE bayraklarından hiçbirini içermiyorsa, DATABASE_OPEN_READWRITE bayrağı kullanılır.

Dosya uzantısı belirtilmezse, ".sqlite" kullanılır.

ENUM_DATABASE_OPEN_FLAGS

ID	Açıklama
DATABASE_OPEN_READONLY	Salt oku

ID	Açıklama
DATABASE_OPEN_READWRITE	Okuma ve yazma için aç
DATABASE_OPEN_CREATE	Gerekirse dosyayı bir diskte oluştur
DATABASE_OPEN_MEMORY	RAM'de veritabanı oluştur
DATABASE_OPEN_COMMON	Dosya tüm terminallerin ortak klasöründedir

Ayrıca bakınız

DatabaseClose

DatabaseClose

Veritabanını kapatır.

```
void  DatabaseClose(
   int  database      // DatabaseOpen'da elde edilen veritabanı tanıtıcı değeri
   );
```

Parametreler

database

[in] DatabaseOpen()'da elde edilen veritabanı tanıtıcı değeri

Geri dönüş değeri

Yok.

Not

DatabaseClose çağrıldıktan sonra, veritabanına yapılan tüm isteklerin tanıtıcı değerleri otomatik olarak kaldırılır ve geçersiz hale gelir.

Eğer tanıtıcı geçersizse, fonksiyon ERR_DATABASE_INVALID_HANDLE hatası verir. Hatayı GetLastError() kullanarak kontrol edebilirsiniz.

Ayrıca bakınız

DatabaseOpen, DatabasePrepare

DatabaseImport

Verileri bir dosyadan tabloya aktarır.

```
long  DatabaseImport(
   int           database,          // DatabaseOpen'da elde edilen veritabanı tanıtıcı
   const string  table,             // veri eklemek için tablo adı
   const string  filename,          // veri aktarmak için dosya adı
   uint          flags,             // bayrak kombinasyonu
   const string  separator,         // veri ayırıcı
   ulong         skip_rows,         // kaç başlangıç dizgesi atlanacak
   const string  skip_comments      // yorumları tanımlayan karakter dizgesi
   );
```

Parametreler

database

[in] DatabaseOpen()'da elde edilen veritabanı tanıtıcı değeri

table

[in] Dosyadaki verilerin ekleneceği tablonun adı.

filename

[in] Veri okumak için CSV dosyası veya ZIP arşivi. Ad, alt dizinler içerebilir ve MQL5\Files klasörüne göre ayarlanır.

flags

[in] ENUM_DATABASE_IMPORT_FLAGS numaralandırmasından bayrak kombinasyonu.

separator

[in] CSV dosyasındaki veri ayırıcı.

skip_rows

[in] Dosyadan veri okunurken atlanacak başlangıç dizgesi sayısı.

skip_comments

[in] Dizgeleri yorum olarak işaretlemek için karakter dizgesi. Bir dizgenin başında *skip_comments*'ten herhangi bir karakter algılanırsa, böyle bir dizge yorum olarak kabul edilir ve içe aktarılmaz.

Geri dönüş değeri

İçe aktarılan dizgelerin sayısını veya bir hata durumunda -1 geri döndürür. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_INVALID_PARAMETER (4003) - tablo adı belirtilmedi (boş dizge veya NULL);
- ERR_DATABASE_INTERNAL (5120) - dahili veritabanı hatası;
- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz veritabanı tanıtıcı değeri.

Not

table adında bir tablo yoksa, otomatik olarak oluşturulur. Oluşturulan tablodaki adlar ve alan türleri, dosya verilerine göre otomatik olarak tanımlanır.

ENUM_DATABASE_IMPORT_FLAGS

Kimlik	Açıklama
DATABASE_IMPORT_HEADER	İlk satırı tablo alanlarının adları olarak içe aktar
DATABASE_IMPORT_CRLF	CRLF'leri dizge sonları olarak kabul et (varsayılan olarak LF'ler dizge sonları olarak kabul edilir)
DATABASE_IMPORT_APPEND	Verileri mevcut tablonun sonuna ekle (varsayılan olarak veriler tablonun üzerine yazılır)
DATABASE_IMPORT_QUOTED_STRINGS	Çift tırnak içerisine alınmış dizge değerlerini içe aktar
DATABASE_IMPORT_COMMON_FOLDER	Dosya, tüm müşteri terminallerinin ortak klasöründe (\Terminal\Common\File) oluşturulur

DatabaseExport örneğinde oluşturulan dosyadan tabloyu içe aktarma örneği:

```
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
void OnStart()
 {
  string csv_filename;
//--- müşteri terminallerinin ortak klasöründen, içe aktarılacak metin dosyalarının ad
  string filenames[];
  if(FileSelectDialog("Tablosunu oluşturmak için CSV dosyası seçin",NULL,
                      "Metin dosyaları (*.csv)|*.csv",
                      FSD_WRITE_FILE|FSD_COMMON_FOLDER,filenames,"veriler.csv")>0)
   {
    //--- seçilen her dosyanın adını göster
    if(ArraySize(filenames)==1)
      csv_filename=filenames[0];
    else
     {
      Print("Dosya seçilirken bilinmeyen hata. Hata kodu ",GetLastError());
      return;
     }
   }
  else
   {
    Print("CSV dosyası seçilmedi");
    return;
   }
```

```
//--- veritabanı oluştur veya aç
  string db_filename="İçeAktarmaTest.sqlite";
  int db=DatabaseOpen(db_filename,DATABASE_OPEN_READWRITE|DATABASE_OPEN_CREATE);
//--- İçeAktarmaTest tablosunun mevcut olup olmadığını kontrol et
  if(DatabaseTableExists(db,"İçeAktarmaTest"))
    {
     //--- İçeAktarmaTest tablosunu kaldır
     if(!DatabaseExecute(db,"DROP TABLE IF EXISTS İçeAktarmaTest"))
      {
       Print("İçeAktarmaTest tablosu kaldırılamadı, hata kodu ",GetLastError());
       DatabaseClose(db);
       return;
      }
    }
  //--- csv dosyasından girdileri İçeAktarmaTest tablosuna içe aktar
  long imported=DatabaseImport(db,"İçeAktarmaTest",csv_filename,DATABASE_IMPORT_HEADER
  if(imported>0)
    {
     Print(imported," satır İçeAktarmaTest tablosuna içe aktarıldı");
     DatabasePrint(db,"SELECT * FROM İçeAktarmaTest",DATABASE_PRINT_NO_INDEX);
    }
  else
    {
     Print("DatabaseImport() başarısız oldu. Hata ",GetLastError());
    }
//--- veritabanı dosyasını kapat ve bunu rapor et
  DatabaseClose(db);
  PrintFormat("%s veritabanı kapatıldı",db_filename);
 }
```

Ayrıca bakınız

DatabaseOpen, DatabasePrint

DatabaseExport

Bir tablo veya SQL isteği yürütme sonucunu bir CSV dosyasına aktarır. Dosya UTF-8 kodlamasında oluşturulur.

```
long  DatabaseExport(
   int           database,          // DatabaseOpen'da elde edilen veritabanı tanıtıc
   const string  table_or_sql,      // tablo adı veya SQL isteği
   const string  filename,          // veri dışa aktarımı için CSV dosyası adı
   uint          flags,             // bayrak kombinasyonu
   const string  separator          // CSV dosyasındaki veri ayırıcı
   );
```

Parametreler

database

[in] DatabaseOpen()'da elde edilen veritabanı tanıtıcı değeri

table_or_sql

[in] Sonuçları belirtilen dosyaya aktarılacak olan tablonun adı veya SQL isteğinin metni.

filename

[in] Veri dışa aktarımı için dosya adı. Yol, MQL5\Files klasörüne göre ayarlanır.

flags

DATABASE_EXPORT_HEADER - alan adlarına sahip bir dizge görüntüle

separator

[in] Veri ayırıcı. NULL belirtilirse, ayırıcı olarak '\t' tablolama karakteri kullanılır. Boş bir dizge "" geçerli bir ayırıcı olarak kabul edilir, ancak elde edilen CSV dosyası tablo olarak okunamaz - bir dizge kümesi olarak kabul edilir.

Geri dönüş değeri

Dışa aktarılan girdilerin sayısını veya bir hata durumunda negatif bir değer geri döndürür. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_INTERNAL_ERROR (4001) - kritik çalışma zamanı hatası;
- ERR_INVALID_PARAMETER (4003) - veritabanı dosyasının yolu boş bir dizge içeriyor ya da uyumsuz bir bayrak kombinasyonu ayarlanmış;
- ERR_NOT_ENOUGH_MEMORY (4004) - yetersiz bellek;
- ERR_FUNCTION_NOT_ALLOWED(4014) - belirtilen veri yoluna izin verilmemektedir;
- ERR_PROGRAM_STOPPED(4022) - işlem iptal edildi (MQL programı durdu);
- ERR_WRONG_FILENAME (5002) - geçersiz dosya adı;
- ERR_TOO_LONG_FILENAME (5003) - dosyanın mutlak yolu maksimum uzunluğu aşıyor;
- ERR_CANNOT_OPEN_FILE(5004) - dosya yazmak için açılamıyor;
- ERR_FILE_WRITEERROR(5026) - dosyaya yazılamıyor;
- ERR_DATABASE_INTERNAL (5120) - dahili veritabanı hatası;
- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz veritabanı tanıtıcı değeri;
- ERR_DATABASE_QUERY_PREPARE(5125) - istek oluşturma hatası;
- ERR_DATABASE_QUERY_NOT_READONLY - salt okunur isteğe izin verilmektedir.

Not

Eğer istek sonuçları dışa aktarılırsa, SQL isteği "SELECT" veya "select" ile başlamalıdır. Başka bir deyişle, SQL isteği veritabanı durumunu değiştiremez, aksi takdirde DatabaseExport() bir hata ile başarısız olur.

Veritabanı dizge değerleri, dönüşüm karakterini ('\r' veya '\r\n') ve ek olarak *separator* parametresinde ayarlanan değer ayırıcı karakterini içerebilir. Bu durumda, 'flags' parametresinde DATABASE_EXPORT_QUOTED_STRINGS bayrağını kullandığınızdan emin olun. Bu bayrak varsa, görüntülenen tüm dizgeler çift tırnak içine alınır. Eğer dizge zaten çift tırnak içerisindeyse, bu çift tırnaklar ikişer adet haline gelir.

ENUM_DATABASE_EXPORT_FLAGS

Kimlik	Açıklama
DATABASE_EXPORT_HEADER	İlk dizgede tablo alanlarının adlarını görüntüle
DATABASE_EXPORT_INDEX	Dizgelerin indekslerini görüntüle
DATABASE_EXPORT_NO_BOM	Dosyanın başına BOM işaretini ekleme (varsayılan olarak BOM işareti eklenir)
DATABASE_EXPORT_CRLF	Dizge sonları için CRLF'ler kullan (varsayılan olarak dizge sonları için LF'ler kullanılır)
DATABASE_EXPORT_APPEND	Verileri mevcut dosyanın sonuna ekle (varsayılan olarak dosyanın üzerine yazılır)
DATABASE_EXPORT_QUOTED_STRINGS	Dizge değerlerini çift tırnak içerisinde görüntüle
DATABASE_EXPORT_COMMON_FOLDER	csv dosyası, tüm müşteri terminallerinin ortak klasöründe (\Terminal\Common\File) oluşturulur

Örnek:

```
input int InpRates=100;
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
void OnStart()
 {
  MqlRates rates[];
//--- çubukları almadan önce başlangıç zamanını hatırla
  ulong start=GetMicrosecondCount();
//--- H1 zaman diliminde son 100 çubuğu iste
  if(CopyRates(Symbol(), PERIOD_H1,1,InpRates,rates)<InpRates)
   {
    Print("CopyRates() başarısız oldu, hata ",GetLastError());
    return;
```

```
    }
  else
    {
     //--- kaç tane çubuk alındı ve bunları almak için ne kadar zaman gerekti
     PrintFormat("%s: CopyRates %d çubuk aldı (%d ms içerisinde)",
                 _Symbol,ArraySize(rates),(GetMicrosecondCount()-start)/1000);
    }
//--- veritabanını depolamak için dosya adını ayarla
  string filename=_Symbol+"_"+EnumToString(PERIOD_H1)+"_"+TimeToString(TimeCurrent())+
  StringReplace(filename,":","-"); // dosya adlarında ":" karakteri kullanılamaz
//--- ortak terminal klasöründe veritabanını aç/oluştur
  int db=DatabaseOpen(filename,DATABASE_OPEN_READWRITE|DATABASE_OPEN_CREATE|DATABASE_C
  if(db==INVALID_HANDLE)
    {
     Print(filename," veritabanının açılışı başarısız oldu, hata ",GetLastError());
     return;
    }
  else
     Print(filename," veritabanı başarıyla açıldı");

//--- Çubuklar tablosunun mevcut olup olmadığını kontrol et
  if(DatabaseTableExists(db,"Çubuklar"))
    {
     //--- Çubuklar tablosunu kaldır
     if(!DatabaseExecute(db,"DROP TABLE IF EXISTS Çubuklar"))
       {
        Print("Çubuklar tablosu kaldırılamadı, hata kodu ",GetLastError());
        DatabaseClose(db);
        return;
       }
    }
//--- Çubuklar tablosunu oluştur
  if(!DatabaseExecute(db,"CREATE TABLE Çubuklar("
                         "Sembol              CHAR(10),"
                         "Zaman                INT NOT NULL,"
                         "Açılış                REAL,"
                         "Yüksek                REAL,"
                         "Düşük                 REAL,"
                         "Kapanış               REAL,"
                         "Tik_Hacim         INT,"
                         "Spread              INT,"
                         "Gerçek_Hacim         INT);"))
    {
     Print(filename," veritabanında Çubuklar tablosu oluşturulamadı, hata ",GetLastErro
     DatabaseClose(db);
     return;
    }
//--- Çubuklar tablosundaki tüm alanların listesini görüntüle
  if(DatabasePrint(db,"PRAGMA TABLE_INFO(Çubuklar)",0)<0)
```

```
    {
     PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(Çubuklar)\") başarısız oldu, hata
     DatabaseClose(db);
     return;
    }
//--- Çubuklar tablosuna çubukları eklemek için parametreli istek oluştur
  string sql="INSERT INTO Çubuklar(Sembol,Zaman,Açılış,Yüksek,Düşük,Kapanış,Tik_Hacmi,
             " VALUES (?1,?2,?3,?4,?5,?6,?7,?8,?9)"; // istek parametreleri
  int request=DatabasePrepare(db,sql);
  if(request==INVALID_HANDLE)
    {
     PrintFormat("DatabasePrepare() başarısız oldu, hata=%d",GetLastError());
     Print("SQL isteği: ",sql);
     DatabaseClose(db);
     return;
    }
//--- ilk istek parametresinin değerini ayarla
  DatabaseBind(request,0,_Symbol);
//--- Çubuklar tablosuna çubukları eklemeden önce başlangıç zamanını hatırla
  start=GetMicrosecondCount();
  DatabaseTransactionBegin(db);
  int total=ArraySize(rates);
  bool request_error=false;
  for(int i=0;i<total;i++)
    {
     //--- girdi eklemeden önce kalan parametrelerin değerlerini ayarla
     ResetLastError();
     if(!DatabaseBind(request,1,rates[i].time))
       {
        PrintFormat("DatabaseBind() başarısız oldu, hata=%d",GetLastError());
        PrintFormat("Çubuk=%d çizgi=%d",i+1,__LINE__);
        request_error=true;
        break;
       }
     //--- önceki DatabaseBind() çağrısı başarılı olduysa, sonraki parametreyi ayarla
     if(!request_error &&! DatabaseBind(request,2,rates[i].open))
       {
        PrintFormat("DatabaseBind() başarısız oldu, hata=%d",GetLastError());
        PrintFormat("Çubuk=%d çizgi=%d",i+1,__LINE__);
        request_error=true;
        break;
       }
     if(!request_error &&! DatabaseBind(request,3,rates[i].high))
       {
        PrintFormat("DatabaseBind() başarısız oldu, hata=%d",GetLastError());
        PrintFormat("Çubuk=%d çizgi=%d",i+1,__LINE__);
        request_error=true;
        break;
       }
```

```
   if(!request_error &&! DatabaseBind(request,4,rates[i].low))
    {
     PrintFormat("DatabaseBind() başarısız oldu, hata=%d",GetLastError());
     PrintFormat("Çubuk=%d çizgi=%d",i+1,__LINE__);
     request_error=true;
     break;
    }
   if(!request_error &&! DatabaseBind(request,5,rates[i].close))
    {
     PrintFormat("DatabaseBind() başarısız oldu, hata=%d",GetLastError());
     PrintFormat("Çubuk=%d çizgi=%d",i+1,__LINE__);
     request_error=true;
     break;
    }
   if(!request_error &&! DatabaseBind(request,6,rates[i].tick_volume))
    {
     PrintFormat("DatabaseBind() başarısız oldu, hata=%d",GetLastError());
     PrintFormat("Çubuk=%d çizgi=%d",i+1,__LINE__);
     request_error=true;
     break;
    }
   if(!request_error &&! DatabaseBind(request,7,rates[i].spread))
    {
     PrintFormat("DatabaseBind() başarısız oldu, hata=%d",GetLastError());
     PrintFormat("Çubuk=%d çizgi=%d",i+1,__LINE__);
     request_error=true;
     break;
    }
   if(!request_error &&! DatabaseBind(request,8,rates[i].real_volume))
    {
     PrintFormat("DatabaseBind() başarısız oldu, hata=%d",GetLastError());
     PrintFormat("Çubuk=%d çizgi=%d",i+1,__LINE__);
     request_error=true;
     break;
    }

   //--- girdiyi eklemek için istek yürüt ve bir hata olup olmadığını kontrol et
   if(!request_error &&! DatabaseRead(request)&&(GetLastError()!=ERR_DATABASE_NO_MORE
    {
     PrintFormat("DatabaseRead() başarısız oldu, hata=%d",GetLastError());
     DatabaseFinalize(request);
     request_error=true;
     break;
    }
   //--- sonraki parametre güncellemesinden önce isteği sıfırla
   if(!request_error &&! DatabaseReset(request))
    {
     PrintFormat("DatabaseReset() başarısız oldu, hata=%d",GetLastError());
     DatabaseFinalize(request);
```

```
      request_error=true;
      break;
     }
   } //--- tüm çubuklardan geçiş tamamlandı

//--- durum
  if(request_error)
   {
    PrintFormat("Çubuklar tablosuna %d çubuk eklenemedi",ArraySize(rates));
    DatabaseTransactionRollback(db);
    DatabaseClose(db);
    return;
   }
  else
   {
    DatabaseTransactionCommit(db);
    PrintFormat("Çubuklar tablosuna %d çubuk eklendi (%d ms içerisinde)",
                ArraySize(rates),(GetMicrosecondCount()-start)/1000);
   }
//--- Çubuklar tablosunu csv dosyasına kaydet
  string csv_filename=Symbol()+".csv";
  long saved=DatabaseExport(db,"SELECT * FROM Çubuklar",csv_filename,DATABASE_EXPORT_F
  if(saved>0)
    Print("Çubuklar tablosu ",Symbol(),".csv dosyasına kaydedildi");
  else
    Print("DatabaseExport() başarısız oldu. Hata ",GetLastError());
//--- veritabanı dosyasını kapat ve bunu rapor et
  DatabaseClose(db);
  PrintFormat("%s veritabanı kapatıldı",filename);
 }
```

Ayrıca bakınız

DatabasePrint, DatabaseImport

DatabasePrint

Uzman Danışmanlar günlüğüne bir tablo veya SQL isteği yürütme sonucu yazdırır.

```
long  DatabasePrint(
   int           database,          // DatabaseOpen'da elde edilen veritabanı tanıtıcı
   const string  table_or_sql,      // tablo veya SQL isteği
   uint          flags              // bayrak kombinasyonu
   );
```

Parametreler

database

[in] DatabaseOpen()'da elde edilen veritabanı tanıtıcı değeri

table_or_sql

[in] Sonuçları Uzman Danışmanlar günlüğünde görüntülenen tablonun adı veya SQL isteğinin metni.

flags

[in] Çıktı biçimlendirmesini tanımlayan bayrakların kombinasyonu. Bayraklar aşağıdaki gibi tanımlanır:

DATABASE_PRINT_NO_HEADER - tablo sütun adlarını (alan adları) gösterme
DATABASE_PRINT_NO_INDEX - dizge numaralarını görüntüleme
DATABASE_PRINT_NO_FRAME - başlık ve veriyi ayıran bir kare görüntüleme
DATABASE_PRINT_STRINGS_RIGHT - dizgeleri sağa hizala.

flags = 0 ise, sütunlar ve dizgeler görüntülenir, başlık ve veri kare ile ayrılırken, dizgeler sola hizalanır.

Geri dönüş değeri

Dışa aktarılan dizgelerin sayısını veya bir hata durumunda -1 geri döndürür. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_INTERNAL_ERROR (4001) - kritik çalışma zamanı hatası;
- ERR_NOT_ENOUGH_MEMORY (4004) - yetersiz bellek;
- ERR_DATABASE_INTERNAL (5120) - dahili veritabanı hatası;
- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz veritabanı tanıtıcı değeri;

Not

Eğer günlük istek sonuçlarını görüntülerse, SQL isteği "SELECT" veya "select" ile başlamalıdır. Başka bir deyişle, SQL isteği veritabanı durumunu değiştiremez, aksi takdirde DatabasePrint() bir hata ile başarısız olur.

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   string filename="departments.sqlite";
```

```
//--- ortak terminal klasöründe veritabanı oluşturma veya açma
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE |DATAB
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }

//--- COMPANY tablosu oluştur
   if(!CreataTableCompany(db))
     {
      DatabaseClose(db);
      return;
     }
//--- DEPARTMENT tablosu oluştur
   if(!CreataTableDepartment(db))
     {
      DatabaseClose(db);
      return;
     }

//--- COMPANY ve DEPARTMENT tablolarındaki tüm alanların listesini görüntüle
   PrintFormat("Try to print request \"PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE_INFO(DE
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(COMPANY);PRAGMA TABLE_INFO(DEPARTMENT)", 0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO()\") failed, error code=%d", GetL
      DatabaseClose(db);
      return;
     }
//--- günlükte COMPANY tablosunu görüntüle
   PrintFormat("Try to print request \"SELECT * from COMPANY\"");
   if(DatabasePrint(db, "SELECT * from COMPANY", 0)<0)
     {
      Print("DatabasePrint failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- COMPANY ve DEPARTMENT tablolarını birleştirmek için istek metni
   string request="SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUTER JOIN DEPARTMENT "
                  "ON COMPANY.ID = DEPARTMENT.EMP_ID";
//--- tabloları birleştirmenin sonucunu göster
   PrintFormat("Try to print request \"SELECT EMP_ID, NAME, DEPT FROM COMPANY LEFT OUT
   if(DatabasePrint(db, request, 0)<0)
     {
      Print("DatabasePrint failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- veritabanını kapat
```

```
   DatabaseClose(db);
  }
/*
Sonuç:
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
*/
//+------------------------------------------------------------------+
//| COMPANY tablosu oluştur                                          |
//+------------------------------------------------------------------+
bool CreateTableCompany(int database)
  {
//--- COMPANY tablosu varsa, sil
   if(DatabaseTableExists(database, "COMPANY"))
     {
      //--- tabloyu sil
      if(!DatabaseExecute(database, "DROP TABLE COMPANY"))
        {
```

```
         Print("Failed to drop table COMPANY with code ", GetLastError());
         return(false);
        }
     }
//--- COMPANY tablosu oluştur
   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: create table COMPANY failed with code ", GetLastError());
      return(false);
     }

//--- COMPANY tablosuna veri gir
   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, 'D
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (6, 'F
                       "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) VALUES (7, 'J
     {
      Print("COMPANY insert failed with code ", GetLastError());
      return(false);
     }
//--- başarı
   return(true);
  }
//+------------------------------------------------------------------+
//| DEPARTMENT tablosu oluştur                                       |
//+------------------------------------------------------------------+
bool CreateTableDepartment(int database)
  {
//--- DEPARTMENT tablosu varsa, sil
   if(DatabaseTableExists(database, "DEPARTMENT"))
     {
      //--- tabloyu sil
      if(!DatabaseExecute(database, "DROP TABLE DEPARTMENT"))
        {
         Print("Failed to drop table DEPARTMENT  with code ", GetLastError());
         return(false);
        }
     }
//--- DEPARTMENT tablosu oluştur
   if(!DatabaseExecute(database, "CREATE TABLE DEPARTMENT ("
                       "ID      INT PRIMARY KEY   NOT NULL,"
```

```
                        "DEPT    CHAR(50)          NOT NULL,"
                        "EMP_ID  INT               NOT NULL);"))
      {
       Print("DB: create table DEPARTMENT failed with code ", GetLastError());
       return(false);
      }

//--- DEPARTMENT tablosuna veri gir
   if(!DatabaseExecute(database, "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (1, '
                        "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (2, 'Engineerin
                        "INSERT INTO DEPARTMENT (ID,DEPT,EMP_ID) VALUES (3, 'Finance',
      {
       Print("DEPARTMENT insert failed with code ", GetLastError());
       return(false);
      }
//--- başarı
   return(true);
  }
//+------------------------------------------------------------------
```

Ayrıca bakınız

DatabaseExport, DatabaseImport

DatabaseTableExists

Veritabanında tablonun varlığını kontrol eder.

```
bool  DatabaseTableExists(
   int     database,      // DatabaseOpen'da elde edilen veritabanı tanıtıcı değeri
   string  table_name     // tablo adı
   );
```

Parametreler

database

[in] DatabaseOpen()'da elde edilen veritabanı tanıtıcı değeri

table_name

[in] Tablo adı.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_INVALID_PARAMETER (4003) - tablo adı belirtilmedi (boş dizge veya NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - istek UTF-8 dizgesine dönüştürülürken hata oluştu;
- ERR_DATABASE_INTERNAL (5120) - dahili veritabanı hatası;
- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz veritabanı tanıtıcı değeri;
- ERR_DATABASE_EXECUTE (5124) - istek yürütme hatası;
- ERR_DATABASE_NO_MORE_DATA (5126) - tablo yok (bir hata değil, normal sona erme).

Ayrıca bakınız

DatabasePrepare, DatabaseFinalize

DatabaseExecute

Belirtilen veritabanına istek yürütür.

```
bool  DatabaseExecute(
   int     database,      // DatabaseOpen'da elde edilen veritabanı tanıtıcı değeri
   string  sql            // SQL isteği
   );
```

Parametreler

database

[in] DatabaseOpen()'da elde edilen veritabanı tanıtıcı değeri

sql

[in] SQL isteği.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_INTERNAL_ERROR (4001) - kritik çalışma zamanı hatası;
- ERR_INVALID_PARAMETER (4003) - sql parametresi boş bir dizge içeriyor;
- ERR_NOT_ENOUGH_MEMORY (4004) - yetersiz bellek;
- ERR_WRONG_STRING_PARAMETER (5040) - istek UTF-8 dizgesine dönüştürülürken hata oluştu;
- ERR_DATABASE_INTERNAL (5120) - dahili veritabanı hatası;
- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz veritabanı tanıtıcı değeri;
- ERR_DATABASE_EXECUTE (5124) - istek yürütme hatası.

Not

DatabasePrepare fonksiyonu bir veritabanına istek gerçekleştirmez. Amacı, istek parametrelerini doğrulamak ve doğrulama sonuçlarına dayanarak DatabaseExecute()'deki SQL isteğini yürütmek için tanıtıcı değerini geri döndürmektir.

Örnek:

```
//--- 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 com
   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 com
   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_
                           "   (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, 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 ", __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);
  }
//+------------------------------------------------------------------+
```

Ayrıca bakınız

DatabasePrepare, DatabaseFinalize

DatabasePrepare

Daha sonra DatabaseRead() kullanılarak yürütülebilen bir istek tanıtıcı değeri oluşturur.

```
int  DatabasePrepare(
   int     database,      // DatabaseOpen'da elde edilen veritabanı tanıtıcı değeri
   string  sql,           // SQL isteği
   uint    ...            // istek parametreleri
   );
```

Parametreler

database

[in] DatabaseOpen()'da elde edilen veritabanı tanıtıcı değeri

sql

[in] ?1,?2,... adında otomatik olarak değiştirilen parametreler içerebilen SQL isteği

...

[in] Otomatik olarak değiştirilen istek parametreleri.

Geri dönüş değeri

Başarılı olursa, fonksiyon SQL isteği için bir tanıtıcı değeri geri döndürür. Aksi takdirde, INVALID_HANDLE geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_INVALID_PARAMETER (4003) - veritabanı dosyasının yolu boş bir dizge içeriyor ya da uyumsuz bir bayrak kombinasyonu ayarlanmış;
- ERR_NOT_ENOUGH_MEMORY (4004) - yetersiz bellek;
- ERR_WRONG_STRING_PARAMETER (5040) - istek UTF-8 dizgesine dönüştürülürken hata oluştu;
- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz veritabanı tanıtıcı değeri;
- ERR_DATABASE_TOO_MANY_OBJECTS (5122) - kabul edilebilir maksimum Database nesnesi sayısı aşıldı;
- ERR_DATABASE_PREPARE (5125) - İstek oluşturma hatası.

Not

DatabasePrepare() fonksiyonu bir veritabanına istek gerçekleştirmez. Amacı, istek parametrelerini doğrulamak ve doğrulama sonuçlarına dayanarak SQL isteğini yürütmek için tanıtıcı değerini geri döndürmektir. İsteğin kendisi DatabaseRead() ilk çağrısı sırasında yürütülür.

Örnek:

```
//--- 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
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);
  }
//+------------------------------------------------------------------+
```

Ayrıca bakınız

DatabaseExecute, DatabaseFinalize

DatabaseReset

DatabasePrepare() çağrıldıktan sonra olduğu gibi isteği sıfırlar.

```
int  DatabaseReset(
   int  request      // DatabasePrepare'da elde edilen istek tanıtıcı değeri
   );
```

Parametreler

request

[in] DatabasePrepare()'da elde edilen isteğin tanıtıcı değeri.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz veritabanı tanıtıcı değeri;
- ERR_DATABASE_ERROR (5601) ile başlayan SQLite hata kodları.

Not

DatabaseReset() fonksiyonu, farklı parametre değerlerine sahip bir isteğin birden çok yürütülmesi için tasarlanmıştır. Örneğin, INSERT komutunu kullanarak tabloya toplu olarak veri eklerken, her girdi için özel bir alan değerleri kümesi oluşturulmalıdır.

DatabasePrepare()'dan farklı olarak, DatabaseReset() çağrısı SQL komutlarını içeren dizgeyi yeni bir istekte derlemez, bu nedenle DatabaseReset(), DatabasePrepare()'dan çok daha hızlı yürütülür.

DatabaseRead() yürütüldükten sonra istek parametresi değerlerinin değiştirilmesi gerekirse, DatabaseReset(), DatabaseBind() fonksiyonları ve/veya DatabaseBindArray() ile birlikte kullanılır. Bu, istek parametrelerinin yeni değerleri ayarlanmadan önce (DatabaseBind/DatabaseBindArray çağrıları bloğundan önce), sıfırlamak için DatabaseReset() çağrılması gerektiği anlamına gelir. Parametrelenmiş isteğin kendisi DatabasePrepare() kullanılarak oluşturulmalıdır.

DatabasePrepare() gibi, DatabaseReset()'de veritabanı isteği yapmaz. DatabaseRead() veya DatabaseReadBind() çağrılırken doğrudan istek yürütme gerçekleştirilir.

DatabaseReset() çağrısı, istekteki parametre değerlerinin sıfırlanmasına yol açmaz (eğer parametre değerleri DatabaseBind()/DatabaseBindArray() çağrılarak ayarlandıysa), yani parametreler değerlerini korur. Böylece, sadece tek bir parametrenin değeri değiştirilebilir. DatabaseReset() çağrıldıktan sonra tüm istek parametrelerini yeniden ayarlamanıza gerek yoktur.

DatabaseFinalize() kullanılarak kaldırılan bir istek tanıtıcı değeri DatabaseReset()'e iletilemez. Bu bir hataya neden olacaktır.

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- veritabanı oluştur veya aç
```

```
   string filename="symbols.sqlite";
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE);
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
   else
      Print("Database: ", filename, " opened successfully");
//--- SYMBOLS tablosu varsa, sil
   if(DatabaseTableExists(db, "SYMBOLS"))
     {
      //--- tabloyu sil
      if(!DatabaseExecute(db, "DROP TABLE SYMBOLS"))
        {
         Print("Failed to drop table SYMBOLS with code ", GetLastError());
         DatabaseClose(db);
         return;
        }
     }
//--- SYMBOLS tablosu oluştur
   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, " create table failed with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- SYMBOLS tablosundaki tüm alanların listesini görüntüle
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(SYMBOLS)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(SYMBOLS)\") failed, error code=%d
      DatabaseClose(db);
      return;
     }

//--- SYMBOLS tablosuna sembol eklemek için parametrelenmiş bir istek oluştur
   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() failed with code=%d", GetLastError());
```

```
      Print("SQL request: ", sql);
      DatabaseClose(db);
      return;
     }

//--- tüm sembolleri gözden geçir ve dosyaları SYMBOLS tablosuna ekle
   int symbols=SymbolsTotal(false);
   bool request_error=false;
   DatabaseTransactionBegin(db);
   for(int i=0; i<symbols; i++)
     {
      //--- sembol eklemeden önce parametrelerin değerlerini ayarla
      ResetLastError();
      string symbol=SymbolName(i, false);
      if(!DatabaseBind(request, 0, symbol))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      //--- önceki DatabaseBind() çağrısı başarılı olduysa, sonraki parametreyi ayarla
      if(!DatabaseBind(request, 1, SymbolInfoString(symbol, SYMBOL_DESCRIPTION)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 2, SymbolInfoString(symbol, SYMBOL_PATH)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 3, SymbolInfoInteger(symbol, SYMBOL_SPREAD)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 4, SymbolInfoDouble(symbol, SYMBOL_POINT)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }
      if(!DatabaseBind(request, 5, SymbolInfoInteger(symbol, SYMBOL_DIGITS)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
```

```
         break;
        }
      if(!DatabaseBind(request, 6, GetSymBolAsJson(symbol)))
        {
         PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, GetLas
         request_error=true;
         break;
        }

      //--- girdiyi eklemek için bir istek yürüt ve bir hata olup olmadığını kontrol e
      if(!DatabaseRead(request)&&(GetLastError()!=ERR_DATABASE_NO_MORE_DATA))
        {
         PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
      else
         PrintFormat("%d: added %s", i+1, symbol);
      //--- sonraki parametre güncellemesinden önce isteği sıfırla
      if(!DatabaseReset(request))
        {
         PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
         DatabaseFinalize(request);
         request_error=true;
         break;
        }
     } //--- tüm semboller gözden geçilerek tamamlandı

//--- işlem durumu
   if(request_error)
     {
      PrintFormat("Table SYMBOLS: failed to add %d symbols", symbols);
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Table SYMBOLS: added %d symbols",symbols);
     }

//--- SYMBOLS tablosunu bir CSV dosyasına kaydet
   string csv_filename="symbols.csv";
   if(DatabaseExport(db, "SELECT * FROM SYMBOLS", csv_filename,
                     DATABASE_EXPORT_HEADER|DATABASE_EXPORT_INDEX|DATABASE_EXPORT_QUOT
      Print("Database: table SYMBOLS saved in ", csv_filename);
   else
```

```
      Print("Database: DatabaseExport(\"SELECT * FROM SYMBOLS\") failed with code", Ge

//--- veritabanı dosyasını kapat ve bunu rapor et
   DatabaseClose(db);
   PrintFormat("Database: %s created and closed", filename);
  }
//+------------------------------------------------------------------+
//| Sembol özelliklerini JSON olarak geri döndürme                   |
//+------------------------------------------------------------------+
string GetSymBolAsJson(string symbol)
  {
//--- girintiler
   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(syn
               "\n"+indent3+"\"ContractSize\":\""+StringFormat("%G",(SymbolInfoDouble
               "\n"+indent3+"\"StopsLevel\":\""+IntegerToString(SymbolInfoInteger(symb
               "\n"+indent3+"\"VolumeMin\":\""+StringFormat("%G",(SymbolInfoDouble(syn
               "\n"+indent3+"\"VolumeMax\":\""+StringFormat("%G",(SymbolInfoDouble(syn
               "\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(symbo
                "\n"+indent2+"}"+
                "\n"+indent1+"]"+
                "\n}";
   return(json);
  }
//+------------------------------------------------------------------+
//| Boşluklardan yapılmış bir girinti oluşturma                      |
//+------------------------------------------------------------------+
string Indent(const int number, const int characters=3)
  {
   int length=number*characters;
   string indent=NULL;
   StringInit(indent, length, ' ');
   return indent;
  }
/*
 Sonuç:
  Database: symbols.sqlite opened successfully
  #| 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
*/
```

Ayrıca bakınız

DatabasePrepare, DatabaseBind, DatabaseBindArray, DatabaseFinalize

DatabaseBind

İstekte bir parametre değeri ayarlar.

```
bool  DatabaseBind(
   int   request,      // DatabasePrepare'da oluşturulan isteğin tanıtıcı değeri
   int   index,        // istekteki parametre indeksi
   T     value         // basit tipteki parametrenin değeri
   );
```

Parametreler

request

[in] DatabasePrepare()'da oluşturulan isteğin tanıtıcı değeri.

index

[in] İstekte değerin ayarlanacağı parametre indeksi. Numaralandırma sıfır ile başlar.

value

[in] Ayarlanacak değer. İzin verilen tipler: bool, char, uchar, short, ushart, int, uint, color, datetime, long, ulong, float, double, string.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_INVALID_PARAMETER (4003) - desteklenmeyen tip;
- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz veritabanı tanıtıcı değeri;
- ERR_DATABASE_NOT_READY (5128) - şu anda bir istekte bulunmak için fonksiyon kullanılamaz. İstek yürütülüyor veya zaten tamamlanmış. DatabaseReset() çağrılmalıdır.

Not

Fonksiyon, bir SQL isteğinin "?" veya "?N" parametrelenebilir değerleri içermesi durumunda kullanılır; burada N, parametre indeksi (birden başlayarak) anlamına gelir. Aynı zamanda, DatabaseBind()'da parametrelerin indekslenmesi sıfırdan başlar.

Örneğin:

```
INSERT INTO table VALUES (?,?,?)
SELECT * FROM table WHERE id=?
```

Bu fonksiyon, DatabasePrepare()'da parametrelenmiş bir istek oluşturulduktan hemen sonra veya istek DatabaseReset() kullanılarak sıfırlandıktan sonra çağrılabilir.

İsteği farklı parametre değerleriyle istediğiniz kadar yürütmek için bu fonksiyonu DatabaseReset() ile birlikte kullanın.

Fonksiyon, basit tipteki parametrelerle çalışacak şekilde tasarlanmıştır. Bir parametrenin bir diziyle karşılaştırılması gerekiyorsa, DatabaseBindArray() fonksiyonunu kullanın.

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
   MqlTick ticks[];
//--- tikleri almadan önce başlangıç zamanını hatırla
   uint start=GetTickCount();
//--- günlük tik geçmişini talep et
   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) failed, error=%d",
                  _Symbol, TimeToString(datetime(from/1000)), TimeToString(datetime(to
      return;
     }
   else
     {
      //--- kaç tane tik alındı ve bunları almak için ne kadar zaman gerekti
      PrintFormat("%s: CopyTicksRange received %d ticks in %d ms (from %s to %s)",
                  _Symbol, ArraySize(ticks), GetTickCount()-start,
                  TimeToString(datetime(from/1000)), TimeToString(datetime(to/1000)));
     }

//--- veritabanını depolamak için dosya adını ayarla
   string filename=_Symbol+" "+TimeToString(datetime(from/1000))+" - "+TimeToString(da
   StringReplace(filename, ":", "."); // dosya adlarında ":" karakteri kullanılamaz
//--- ortak terminal klasöründe veritabanını aç/oluştur
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE | DATA
   if(db==INVALID_HANDLE)
     {
      Print("Database: ", filename, " open failed with code ", GetLastError());
      return;
     }
   else
      Print("Database: ", filename, " opened successfully");

//--- TICKS tablosu oluştur
   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, " create table TICKS failed with code ", GetLastError())
      DatabaseClose(db);
      return;
     }
//--- TICKS tablosundaki tüm alanların listesini görüntüle
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(TICKS)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(TICKS)\") failed, error code=%d a
      DatabaseClose(db);
      return;
     }
//--- TICKS tablosuna tik eklemek için parametrelenmiş bir istek oluştur
   string sql="INSERT INTO TICKS (SYMBOL,TIME,BID,ASK,LAST,VOLUME,TIME_MSC,VOLUME_REAL
              " VALUES (?1,?2,?3,?4,?5,?6,?7,?8)"; // request parameters
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
     {
      PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
      Print("SQL request: ", sql);
      DatabaseClose(db);
      return;
     }
//--- ilk istek parametresinin değerini ayarla
   DatabaseBind(request, 0, _Symbol);
//--- TICKS tablosuna tikleri eklemeden önce başlangıç zamanını hatırla
   start=GetTickCount();
   DatabaseTransactionBegin(db);
   int total=ArraySize(ticks);
   bool request_error=false;
   for(int i=0; i<total; i++)
     {
      //--- girdi eklemeden önce kalan parametrelerin değerlerini ayarla
      ResetLastError();
      if(!DatabaseBind(request, 1, ticks[i].time))
        {
         PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
         PrintFormat("Tick #%d line=%d", i+1, __LINE__);
         request_error=true;
         break;
        }
      //--- önceki DatabaseBind() çağrısı başarılı olduysa, sonraki parametreyi ayarla
      if(!request_error && !DatabaseBind(request, 2, ticks[i].bid))
        {
         PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
         PrintFormat("Tick #%d line=%d", i+1, __LINE__);
         request_error=true;
         break;
        }
      if(!request_error && !DatabaseBind(request, 3, ticks[i].ask))
```

```
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
    if(!request_error && !DatabaseBind(request, 4, ticks[i].last))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
    if(!request_error && !DatabaseBind(request, 5, ticks[i].volume))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
    if(!request_error && !DatabaseBind(request, 6, ticks[i].time_msc))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }
    if(!request_error && !DatabaseBind(request, 7, ticks[i].volume_real))
      {
       PrintFormat("DatabaseBind() failed with code=%d", GetLastError());
       PrintFormat("Tick #%d line=%d", i+1, __LINE__);
       request_error=true;
       break;
      }

    //--- girdiyi eklemek için bir istek yürüt ve bir hata olup olmadığını kontrol e
    if(!request_error && !DatabaseRead(request) && (GetLastError()!=ERR_DATABASE_NO_
      {
       PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
       DatabaseFinalize(request);
       request_error=true;
       break;
      }
    //--- sonraki parametre güncellemesinden önce isteği sıfırla
    if(!request_error && !DatabaseReset(request))
      {
       PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
       DatabaseFinalize(request);
       request_error=true;
```

```
         break;
        }
     } //--- tüm tikler gözden geçilerek tamamlandı

//--- işlem durumu
   if(request_error)
     {
      PrintFormat("Table TICKS: failed to add %d ticks ", ArraySize(ticks));
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Table TICKS: added %d ticks in %d ms",
                  ArraySize(ticks), GetTickCount()-start);
     }

//--- veritabanı dosyasını kapat ve bunu rapor et
   DatabaseClose(db);
   PrintFormat("Database: %s created and closed", filename);
  }
/*
 Sonuç:
  EURUSD: CopyTicksRange received 268061 ticks in 47 ms (from 2020.03.18 12:40 to 202(
  Database: EURUSD 2020.03.18 12.40 - 2020.03.19 12.40.sqlite opened successfully
  #| 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
*/
```

Ayrıca bakınız

DatabasePrepare, DatabaseReset, DatabaseRead, DatabaseBindArray

DatabaseBindArray

Diziyi parametre değeri olarak ayarlar.

```
bool  DatabaseBind(
   int   request,       // DatabasePrepare'da oluşturulan isteğin tanıtıcı değeri
   int   index,         // istekteki parametre indeksi
   T&    array[]        // dizi olarak parametre değeri
   );
```

Parametreler

request

[in] DatabasePrepare()'da oluşturulan isteğin tanıtıcı değeri.

index

[in] İstekte değerin ayarlanacağı parametre indeksi. Numaralandırma sıfır ile başlar.

array[]

[in] İstek parametresi değeri olarak ayarlanacak dizi.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_INVALID_PARAMETER (4003) - desteklenmeyen tip;
- ERR_ARRAY_BAD_SIZE (4011) - bayt cinsinden dizi boyutu INT_MAX değerini aşıyor;
- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz veritabanı tanıtıcı değeri;
- ERR_DATABASE_NOT_READY (5128) - şu anda bir istekte bulunmak için fonksiyon kullanılamaz (istek yürütülüyor veya zaten tamamlandı, DatabaseReset çağrılmalıdır).

Not

Fonksiyon, bir SQL isteğinin "?" veya "?N" parametrelenebilir değerleri içermesi durumunda kullanılır; burada N, parametre indeksi (birden başlayarak) anlamına gelir. Aynı zamanda, DatabaseBindArray()'de parametrelerin endekslenmesi sıfırdan başlar.

Örneğin:

```
INSERT INTO table VALUES (?,?,?)
```

Bu fonksiyon, DatabasePrepare()'da parametrelenmiş bir istek oluşturulduktan hemen sonra veya istek DatabaseReset() kullanılarak sıfırlandıktan sonra çağrılabilir.

İsteği farklı parametre değerleriyle istediğiniz kadar yürütmek için bu fonksiyonu DatabaseReset() ile birlikte kullanın.

Örnek:

```
//+------------------------------------------------------------------+
//| Script programı başlatma fonksiyonu                              |
//+------------------------------------------------------------------+
void OnStart()
  {
```

```
//--- DAT uzantılı dosyaları seçmek için iletişim kutusunu aç
   string selected_files[];
   if(!FileSelectDialog("İndirilecek dosyaları seçin", NULL,
                        "Data files (*.dat)|*.dat|All files (*.*)|*.*",
                        FSD_ALLOW_MULTISELECT, selected_files, "tester.dat")>0)
     {
      Print("Files not selected. Exit");
      return;
     }
//--- dosyaların boyutunu elde et
   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];
        }
     }
//--- toplam dosya boyutunu kontrol et
   if(total_size==0)
     {
      PrintFormat("Total files size is 0. Exit");
      return;
     }

//--- ortak terminal klasöründe veritabanı oluştur veya aç
   string filename="dat_files.sqlite";
   int db=DatabaseOpen(filename, DATABASE_OPEN_READWRITE | DATABASE_OPEN_CREATE);
   if(db==INVALID_HANDLE)
     {
      Print("DB: ", filename, " open failed with code ", GetLastError());
      return;
     }
   else
      Print("Database: ", filename, " opened successfully");
//--- FILES tablosu varsa, sil
   if(DatabaseTableExists(db, "FILES"))
     {
      //--- tabloyu sil
      if(!DatabaseExecute(db, "DROP TABLE FILES"))
```

```
         {
          Print("Failed to drop table FILES with code ", GetLastError());
          DatabaseClose(db);
          return;
         }
      }
//--- FILES tablosu oluştur
   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: failed to create table FILES with code ", GetLastError());
      DatabaseClose(db);
      return;
     }
//--- FILES tablosundaki tüm alanların listesini görüntüle
   if(DatabasePrint(db, "PRAGMA TABLE_INFO(FILES)", 0)<0)
     {
      PrintFormat("DatabasePrint(\"PRAGMA TABLE_INFO(FILES)\") failed, error code=%d a
      DatabaseClose(db);
      return;
     }

//--- FILES tablosuna dosya eklemek için parametrelenmiş bir istek oluştur
   string sql="INSERT INTO FILES (NAME,SIZE,PERCENT_SIZE,DATA)"
              " VALUES (?1,?2,?3,?4);"; // istek parametreleri
   int request=DatabasePrepare(db, sql);
   if(request==INVALID_HANDLE)
     {
      PrintFormat("DatabasePrepare() failed with code=%d", GetLastError());
      Print("SQL request: ", sql);
      DatabaseClose(db);
      return;
     }

//--- tüm dosyaları gözden geçir ve dosyaları FILES tablosuna ekle
   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) failed with code %d", selected_files[i], Ge
       continue;
      }

   count++;
   //--- dosyayı tabloya eklemeden önce parametrelerin değerlerini ayarla
   if(!DatabaseBind(request, 0, selected_files[i]))
      {
       PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, Get
       request_error=true;
       break;
      }
   if(!DatabaseBind(request, 1, size))
      {
       PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, Get
       request_error=true;
       break;
      }
   if(!DatabaseBind(request, 2, double(size)*100./total_size))
      {
       PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, Get
       request_error=true;
       break;
      }
   if(!DatabaseBindArray(request, 3, data))
      {
       PrintFormat("DatabaseBind() failed at line %d with code=%d", __LINE__, Get
       request_error=true;
       break;
      }
   //--- girdiyi eklemek için bir istek yürüt ve bir hata olup olmadığını kontro
   if(!DatabaseRead(request)&&(GetLastError()!=ERR_DATABASE_NO_MORE_DATA))
      {
       PrintFormat("DatabaseRead() failed with code=%d", GetLastError());
       DatabaseFinalize(request);
       request_error=true;
       break;
      }
   else
       PrintFormat("%d. %s: %d bytes", count, selected_files[i],size);
   //--- sonraki parametre güncellemesinden önce isteği sıfırla
   if(!DatabaseReset(request))
      {
       PrintFormat("DatabaseReset() failed with code=%d", GetLastError());
       DatabaseFinalize(request);
       request_error=true;
       break;
      }
```

```
         }
      }
//--- işlem durumu
   if(request_error)
     {
      PrintFormat("Table FILES: failed to add %d files", count);
      DatabaseTransactionRollback(db);
      DatabaseClose(db);
      return;
     }
   else
     {
      DatabaseTransactionCommit(db);
      PrintFormat("Table FILES: added %d files", count);
     }

//--- veritabanı dosyasını kapat ve bunu rapor et
   DatabaseClose(db);
   PrintFormat("Database: %s created and closed", filename);
  }
```

Ayrıca bakınız

DatabasePrepare, DatabaseReset, DatabaseRead, DatabaseBind

DatabaseRead

İstek sonucunda bir sonraki girdiye gider.

```
bool  DatabaseRead(
   int     database,      // DatabaseOpen'da elde edilen veritabanı tanıtıcı değeri
   string  table_name     // tablo adı
   );
```

Parametreler

database

[in] DatabaseOpen()'da elde edilen veritabanı tanıtıcı değeri

table_name

[in] Tablo adı.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_INVALID_PARAMETER (4003) - tablo adı belirtilmedi (boş dizge veya NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - istek UTF-8 dizgesine dönüştürülürken hata oluştu;
- ERR_DATABASE_INTERNAL (5120) - dahili veritabanı hatası;
- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz veritabanı tanıtıcı değeri;
- ERR_DATABASE_EXECUTE (5124) - istek yürütme hatası;
- ERR_DATABASE_NO_MORE_DATA (5126) - tablo yok (bir hata değil, normal sona erme).

Ayrıca bakınız

DatabasePrepare, DatabaseReadBind

DatabaseReadBind

Bir sonraki kayda gider ve verileri yapıya okur.

```
bool  DatabaseReadBind(
   int    request,     // DatabasePrepare'da oluşturulan isteğin tanıtıcı değeri
   void&  struct_object       // kaydı okumak için yapıya olan referans
   );
```

Parametreler

request

[in] DatabasePrepare()'da oluşturulan isteğin tanıtıcı değeri.

struct_object

[out] Geçerli kayıttaki verilerin okunacağı yapıya olan referans. Yapı üye olarak yalnızca sayısal tiplere ve/veya dizgelere (dizilere izin verilmez) sahip olmalı ve ayrıca yapı türetilmiş olamaz.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_INVALID_PARAMETER (4003) - tablo adı belirtilmedi (boş dizge veya NULL);
- ERR_WRONG_STRING_PARAMETER (5040) - istek UTF-8 dizgesine dönüştürülürken hata oluştu;
- ERR_DATABASE_INTERNAL (5120) - dahili veritabanı hatası;
- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz veritabanı tanıtıcı değeri;
- ERR_DATABASE_EXECUTE (5124) - istek yürütme hatası;
- ERR_DATABASE_NO_MORE_DATA (5126) - tablo yok (bir hata değil, normal sona erme).

Not

struct_object yapısındaki alan sayısı DatabaseColumnsCount() değerini geçmemelidir. Eğer *struct_object* yapısındaki alan sayısı, kayıttaki alan sayısından azsa, kısmi okuma gerçekleştirilir. Kalan veriler, ilişkili DatabaseColumnText(), DatabaseColumnInteger() ve diğer fonksiyonlar kullanılarak açıkça elde edilebilir.

Örnek:

```
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
*/
```

Ayrıca bakınız

DatabasePrepare, DatabaseRead

DatabaseFinalize

DatabasePrepare()'da oluşturulan isteği kaldırır.

```
void  DatabaseFinalize(
   int  request      // DatabasePrepare'da elde edilen istek tanıtıcı değeri
   );
```

Parametreler

database

[in] DatabasePrepare()'da elde edilen istek tanıtıcı değeri.

Geri dönüş değeri

Yok.

Not

Eğer tanıtıcı geçersizse, fonksiyon ERR_DATABASE_INVALID_HANDLE hatası verir. Hatayı GetLastError() kullanarak kontrol edebilirsiniz.

Ayrıca bakınız

DatabasePrepare, DatabaseExecute

DatabaseTransactionBegin

İşlem yürütmeyi başlatır.

```
bool  DatabaseTransactionBegin(
   int  database      // DatabaseOpen'da elde edilen veritabanı tanıtıcı değeri
   );
```

Parametreler

database

[in] DatabaseOpen()'da elde edilen veritabanı tanıtıcı değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_INTERNAL_ERROR (4001) - kritik çalışma zamanı hatası;
- ERR_INVALID_PARAMETER (4003) - sql parametresi boş bir dizge içeriyor;
- ERR_NOT_ENOUGH_MEMORY (4004) - yetersiz bellek;
- ERR_WRONG_STRING_PARAMETER (5040) - istek UTF-8 dizgesine dönüştürülürken hata oluştu;
- ERR_DATABASE_INTERNAL (5120) - dahili veritabanı hatası;
- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz veritabanı tanıtıcı değeri;
- ERR_DATABASE_EXECUTE (5124) - istek yürütme hatası.

Not

Bir işlem yürütülmeden önce DatabaseTransactionBegin() fonksiyonu çağrılmalıdır. Herhangi bir işlem DatabaseTransactionBegin() çağrısı ile başlamalı ve DatabaseTransactionCommit() çağrısı ile sona ermelidir.

Örnek:

```
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- create the file name
   string filename=AccountInfoString(ACCOUNT_SERVER) +"_"+IntegerToString(AccountInfol
//--- 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/DatabaseTransactio
   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 m
```

```
   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,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 #%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);
  }
//+------------------------------------------------------------------+
```

Ayrıca bakınız

DatabaseExecute, DatabasePrepare, DatabaseTransactionCommit, DatabaseTransactionRollback

DatabaseTransactionCommit

İşlem yürütmeyi tamamlar.

```
bool  DatabaseTransactionCommit(
   int  database      // DatabaseOpen'da elde edilen veritabanı tanıtıcı değeri
   );
```

Parametreler

database

[in] DatabaseOpen()'da elde edilen veritabanı tanıtıcı değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_INTERNAL_ERROR (4001) - kritik çalışma zamanı hatası;
- ERR_INVALID_PARAMETER (4003) - sql parametresi boş bir dizge içeriyor;
- ERR_NOT_ENOUGH_MEMORY (4004) - yetersiz bellek;
- ERR_WRONG_STRING_PARAMETER (5040) - istek UTF-8 dizgesine dönüştürülürken hata oluştu;
- ERR_DATABASE_INTERNAL (5120) - dahili veritabanı hatası;
- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz veritabanı tanıtıcı değeri;
- ERR_DATABASE_EXECUTE (5124) - istek yürütme hatası.

Not

DatabaseTransactionCommit() fonksiyonu, DatabaseBeginTransaction() fonksiyonunu çağrıldıktan sonra yürütülen tüm işlemleri tamamlar. Herhangi bir işlem DatabaseTransactionBegin() çağrısı ile başlamalı ve DatabaseTransactionCommit() çağrısı ile sona ermelidir.

Ayrıca bakınız

DatabaseExecute, DatabasePrepare, DatabaseTransactionBegin, DatabaseTransactionRollback

DatabaseTransactionRollback

İşlemleri geri alır.

```
bool  DatabaseTransactionRollback(
   int  database      // DatabaseOpen'da elde edilen veritabanı tanıtıcı değeri
   );
```

Parametreler

database

[in] DatabaseOpen()'da elde edilen veritabanı tanıtıcı değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_INTERNAL_ERROR (4001) - kritik çalışma zamanı hatası;
- ERR_INVALID_PARAMETER (4003) - sql parametresi boş bir dizge içeriyor;
- ERR_NOT_ENOUGH_MEMORY (4004) - yetersiz bellek;
- ERR_WRONG_STRING_PARAMETER (5040) - istek UTF-8 dizgesine dönüştürülürken hata oluştu;
- ERR_DATABASE_INTERNAL (5120) - dahili veritabanı hatası;
- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz veritabanı tanıtıcı değeri;
- ERR_DATABASE_EXECUTE (5124) - istek yürütme hatası.

Not

DatabaseTransactionRollback() çağrısı, DatabaseTransactionBegin() fonksiyonu çağrıldıktan sonra yürütülen tüm işlemleri iptal eder. DatabaseTransactionRollback() fonksiyonu, bir işlem yürütülürken hata oluşması durumunda veritabanındaki değişiklikleri geri almak için gereklidir.

Ayrıca bakınız

DatabaseExecute, DatabasePrepare, DatabaseTransactionBegin, DatabaseTransactionCommit

DatabaseColumnsCount

İstekteki alan sayısını elde eder.

```
int  DatabaseColumnsCount(
   int     request      // DatabasePrepare'da elde edilen istek tanıtıcı değeri
   );
```

Parametreler

request

[in] DatabasePrepare()'da elde edilen istek tanıtıcı değeri.

Geri dönüş değeri

Alan sayısı veya hata durumunda -1. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz istek tanıtıcı değeri.

Not

DatabasePrepare()'da oluşturulan bir isteğin alan sayısını elde etmek için DatabaseRead() fonksiyonunu çağırmaya gerek yoktur. Kalan DatabaseColumnXXX() fonksiyonları için DatabaseRead() önceden çağrılmalıdır.

Ayrıca bakınız

DatabasePrepare, DatabaseFinalize, DatabaseClose

DatabaseColumnName

Alan adını indekse göre elde eder.

```
bool  DatabaseColumnName(
   int       request,     // DatabasePrepare'da elde edilen istek tanıtıcı değeri
   int       column,      // istekteki alan indeksi
   string&   name         // tablo adını elde etmek için değişkene olan referans
   );
```

Parametreler

request

[in] DatabasePrepare()'da elde edilen istek tanıtıcı değeri.

column

[in] İstekteki alan indeksi. Alan numaralandırma sıfırdan başlar ve DatabaseColumnsCount() - 1 değerini aşamaz.

name

[out] Alan değerini yazmak için değişken.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz istek tanıtıcı değeri;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' indeksi DatabaseColumnsCount() -1 değerini aşıyor.

Not

Değer, yalnızca 'istek' için önceden en az bir DatabaseRead() çağrısı yapılmışsa elde edilebilir.

Ayrıca bakınız

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType

DatabaseColumnType

Alan tipini indekse göre elde eder.

```
ENUM_DATABASE_FIELD_TYPE  DatabaseColumnType(
   int  request,     // DatabasePrepare'da elde edilen istek tanıtıcı değeri
   int  column       // istekteki alan indeksi
   );
```

Parametreler

request

[in] DatabasePrepare()'da elde edilen istek tanıtıcı değeri.

column

[in] İstekteki alan indeksi. Alan numaralandırma sıfırdan başlar ve DatabaseColumnsCount() - 1 değerini aşamaz.

Geri dönüş değeri

ENUM_DATABASE_FIELD_TYPE listesinden alan tipi geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz istek tanıtıcı değeri;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' indeksi DatabaseColumnsCount() -1 değerini aşıyor.

Not

Değer, yalnızca 'istek' için önceden en az bir DatabaseRead() çağrısı yapılmışsa elde edilebilir.

ENUM_DATABASE_FIELD_TYPE

ID	Açıklama
DATABASE_FIELD_TYPE_INVALID	Tip elde edilirken hata oluştu, hata kodu int GetLastError() kullanılarak öğrenilebilir
DATABASE_FIELD_TYPE_INTEGER	Tam sayı tipi
DATABASE_FIELD_TYPE_FLOAT	Reel tip
DATABASE_FIELD_TYPE_TEXT	Dizge tipi
DATABASE_FIELD_TYPE_BLOB	İkili tip
DATABASE_FIELD_TYPE_NULL	Özel NULL tipi

Ayrıca bakınız

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnName

DatabaseColumnSize

Alan boyutunu bayt cinsinden elde eder.

```
int  DatabaseColumnSize(
   int  request,     // DatabasePrepare'da elde edilen istek tanıtıcı değeri
   int  column       // istekteki alan indeksi
   );
```

Parametreler

request

[in] DatabasePrepare()'da elde edilen istek tanıtıcı değeri.

column

[in] İstekteki alan indeksi. Alan numaralandırma sıfırdan başlar ve DatabaseColumnsCount() - 1 değerini aşamaz.

Geri dönüş değeri

Başarılı olursa, bayt cinsinden alan boyutu geri döndürülür, aksi takdirde -1 geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz istek tanıtıcı değeri;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' indeksi DatabaseColumnsCount() -1 değerini aşıyor.

Not

Değer, yalnızca 'istek' için önceden en az bir DatabaseRead() çağrısı yapılmışsa elde edilebilir.

Ayrıca bakınız

DatabasePrepare, DatabaseColumnBlob, DatabaseColumnsCount, DatabaseColumnName, DatabaseColumnType

DatabaseColumnText

Alan değerini geçerli kayıttan dizge olarak elde eder.

```
bool  DatabaseColumnText(
   int      request,     // DatabasePrepare'da elde edilen istek tanıtıcı değeri
   int      column,      // istekteki alan indeksi
   string&  value        // değeri elde etmek için değişkene olan referans
   );
```

Parametreler

request

[in] DatabasePrepare()'da elde edilen istek tanıtıcı değeri.

column

[in] İstekteki alan indeksi. Alan numaralandırma sıfırdan başlar ve DatabaseColumnsCount() - 1 değerini aşamaz.

value

[out] Alan değerini yazmak için değişkene olan referans.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz istek tanıtıcı değeri;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' indeksi DatabaseColumnsCount() -1 değerini aşıyor.

Not

Değer, yalnızca 'istek' için önceden en az bir DatabaseRead() çağrısı yapılmışsa elde edilebilir.

Sonraki kayıttaki değeri okumak için, önceden DatabaseRead()'i çağırın.

Ayrıca bakınız

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

DatabaseColumnInteger

Geçerli kayıttan int tipi değeri elde eder.

```
bool  DatabaseColumnInteger(
   int      request,     // DatabasePrepare'da elde edilen istek tanıtıcı değeri
   int      column,      // istekteki alan indeksi
   int&     value        // değeri elde etmek için değişkene olan referans
   );
```

Parametreler

request

[in] DatabasePrepare()'da elde edilen istek tanıtıcı değeri.

column

[in] İstekteki alan indeksi. Alan numaralandırma sıfırdan başlar ve DatabaseColumnsCount() - 1 değerini aşamaz.

value

[out] Alan değerini yazmak için değişkene olan referans.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz istek tanıtıcı değeri;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' indeksi DatabaseColumnsCount() -1 değerini aşıyor.

Not

Değer, yalnızca 'istek' için önceden en az bir DatabaseRead() çağrısı yapılmışsa elde edilebilir.

Sonraki kayıttaki değeri okumak için, önceden DatabaseRead()'i çağırın.

Ayrıca bakınız

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

DatabaseColumnLong

Geçerli kayıttan long tipi değeri elde eder.

```
bool  DatabaseColumnLong(
   int       request,     // DatabasePrepare'da elde edilen istek tanıtıcı değeri
   int       column,      // istekteki alan indeksi
   long&     value        // değeri elde etmek için değişkene olan referans
   );
```

Parametreler

request

[in] DatabasePrepare()'da elde edilen istek tanıtıcı değeri.

column

[in] İstekteki alan indeksi. Alan numaralandırma sıfırdan başlar ve DatabaseColumnsCount() - 1 değerini aşamaz.

value

[out] Alan değerini yazmak için değişkene olan referans.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz istek tanıtıcı değeri;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' indeksi DatabaseColumnsCount() -1 değerini aşıyor.

Not

Değer, yalnızca 'istek' için önceden en az bir DatabaseRead() çağrısı yapılmışsa elde edilebilir.

Sonraki kayıttaki değeri okumak için, önceden DatabaseRead()'i çağırın.

Ayrıca bakınız

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

DatabaseColumnDouble

Geçerli kayıttan double tipi değeri elde eder.

```
bool  DatabaseColumnDouble(
   int      request,     // DatabasePrepare'da elde edilen istek tanıtıcı değeri
   int      column,      // istekteki alan indeksi
   long&    value        // değeri elde etmek için değişkene olan referans
   );
```

Parametreler

request

[in] DatabasePrepare()'da elde edilen istek tanıtıcı değeri.

column

[in] İstekteki alan indeksi. Alan numaralandırma sıfırdan başlar ve DatabaseColumnsCount() - 1 değerini aşamaz.

value

[out] Alan değerini yazmak için değişkene olan referans.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz istek tanıtıcı değeri;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' indeksi DatabaseColumnsCount() -1 değerini aşıyor.

Not

Değer, yalnızca 'istek' için önceden en az bir DatabaseRead() çağrısı yapılmışsa elde edilebilir.

Sonraki kayıttaki değeri okumak için, önceden DatabaseRead()'i çağırın.

Ayrıca bakınız

DatabasePrepare, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

DatabaseColumnBlob

Alan değerini geçerli kayıttan dizi olarak elde eder.

```
bool  DatabaseColumnBlob(
   int       request,      // DatabasePrepare'da elde edilen istek tanıtıcı değeri
   int       column,       // istekteki alan indeksi
   void&     data[]        // değeri elde etmek için değişkene olan referans
   );
```

Parametreler

request

[in] DatabasePrepare()'da elde edilen istek tanıtıcı değeri.

column

[in] İstekteki alan indeksi. Alan numaralandırma sıfırdan başlar ve DatabaseColumnsCount() - 1 değerini aşamaz.

data[]

[out] Alan değerini yazmak için diziye olan referans.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false olarak geri döner. Hata kodunu almak için GetLastError() kullanın, olası yanıtlar şunlardır:

- ERR_DATABASE_INVALID_HANDLE (5121) - geçersiz istek tanıtıcı değeri;
- ERR_DATABASE_NO_MORE_DATA (5126) - 'column' indeksi DatabaseColumnsCount() -1 değerini aşıyor.

Not

Değer, yalnızca 'istek' için önceden en az bir DatabaseRead() çağrısı yapılmışsa elde edilebilir.

Sonraki kayıttaki değeri okumak için, önceden DatabaseRead()'i çağırın.

Ayrıca bakınız

DatabasePrepare, DatabaseColumnSize, DatabaseColumnsCount, DatabaseColumnType, DatabaseColumnName

DirectX ile çalışma

DirectX 11 fonksiyonları ve gölgelendiricileri, doğrudan fiyat grafiğinde 3D görselleştirme için tasarlanmıştır.

3D grafikler oluşturmak için gerekli görüntü boyutuna sahip bir grafik içeriği (DXContextCreate) gerekir. Ayrıca, köşe ve indeks tamponları (DXBufferCreate) hazırlamak, ilaveten de köşe ve piksel gölgelendiricileri (DXShaderCreate) oluşturmak gerekir. Bu, grafikleri renkli görüntülemek için yeterlidir.

İleri grafik seviyesi, gölgelendiricilere ek render alma parametreleri iletmek için girdilerin (DXInputSet) varlığına ihtiyaç duyar. Bu, kamera ve 3D nesne konumlarının ayarlanmasına, ışık kaynaklarının tanımlanmasına ve ayrıca fare ve klavye kontrolünün gerçekleştirilmesine olanak sağlar.

Böylece, dahili MQL5 fonksiyonları, üçüncü parti araçlara gerek kalmadan doğrudan MetaTrader 5'de animasyonlu 3D grafikler oluşturmanıza olanak tanır. Grafik kartı, fonksiyonların çalışması için DX 11 ve Shader Model 5.0'ı desteklemelidir.

Kütüphaneyle çalışmaya başlamak için MetaTrader 5'te DirectX'i kullanarak 3D grafik oluşturma makalesini okuyun.

Fonksiyon	Eylem
DXContextCreate	Belirli bir boyuttaki kareleri render almak için bir grafik içeriği oluşturur
DXContextSetSize	DXContextCreate()'de oluşturulan grafik içeriğinin kare boyutunu değiştirir
DXContextSetSize	DXContextCreate()'de oluşturulan grafik içeriğinin kare boyutunu elde eder
DXContextClearColors	Render alma tamponu için tüm pikselleri belirtilen renge ayarlar
DXContextClearDepth	Derinlik tamponunu temizler
DXContextGetColors	Grafik içeriğinden belirtilen boyutta ve ofsette bir görüntü elde eder
DXContextGetDepth	Oluşturulan bir karenin derinlik tamponunu elde eder
DXBufferCreate	Veri dizisine dayalı olarak belirtilen tipte bir tampon oluşturur
DXTextureCreate	Aktarılan görüntüden kesilmiş belirli bir boyuttaki dikdörtgenden 2D bir doku oluşturur
DXInputCreate	Gölgelendirici girdilerini oluşturur
DXInputSet	Gölgelendirici girdilerini ayarlar
DXShaderCreate	Belirtilen tipte bir gölgelendirici oluşturur
DXShaderSetLayout	Köşe gölgelendirici için köşe düzenini ayarlar
DXShaderInputsSet	Gölgelendirici girdilerini ayarlar

Fonksiyon	Eylem
DXShaderTexturesSet	Gölgelendirici dokularını ayarlar
DXDraw	DXBufferSet()'te ayarlanan köşe tamponunun köşe noktalarını render alır
DXDrawIndexed	DXBufferSet() indeks tamponu tarafından tanımlanan grafik ilkellerini oluşturur
DXPrimiveTopologySet	DXDrawIndexed()'i kullanarak render almak için ilkellerin tipini ayarlar
DXBufferSet	Geçerli render işlemi için bir tampon ayarlar
DXShaderSet	Render alma için gölgelendirici ayarlar
DXHandleType	Tanıtıcı değer tipini geri döndürür
DXRelease	Tanıtıcı değeri serbest bırakır

DXContextCreate

Belirli bir boyuttaki kareleri render almak için bir grafik içeriği oluşturur.

```
int  DXContextCreate(
   uint  width,      // piksel cinsinden genişlik
   uint  height      // piksel cinsinden yükseklik
   );
```

Parametreler

width

[in] Piksel cinsinden kare genişliği.

height

[in] Piksel cinsinden kare yüksekliği.

Geri dönüş değeri

Oluşturulan içeriğin tanıtıcı değeri ya da hata durumunda INVALID_HANDLE. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

DXBufferCreate, DXInputCreate, DXShaderCreate ve DXTextureCreate fonksiyonları kullanılarak oluşturulan tüm grafik nesneleri yalnızca oluşturuldukları grafik içeriğinde kullanılabilir.

Kare boyutu daha sonra DXContextSetSize() ile değiştirilebilir.

Artık kullanılmayan bir tanıtıcı değer, DXRelease() fonksiyonu tarafından açıkça serbest bırakılmalıdır.

DXContextSetSize

DXContextCreate()'de oluşturulan grafik içeriğinin kare boyutunu değiştirir.

```
bool  DXContextSetSize(
   int    context,      // grafik içeriği tanıtıcı değeri
   uint&  width,        // piksel cinsinden genişlik
   uint&  height        // piksel cinsinden yükseklik
   );
```

Parametreler

context

[in] DXContextCreate()'te oluşturulan grafik içeriği için tanıtıcı değer.

width

[in] Piksel cinsinden kare genişliği.

height

[in] Piksel cinsinden kare yüksekliği.

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

Grafik içeriğinin kare boyutu yalnızca karenin render işlemlerinin arasında değiştirilmelidir.

DXContextGetSize

DXContextCreate()'de oluşturulan grafik içeriğinin kare boyutunu elde eder.

```
bool  DXContextGetSize(
   int    context,       // grafik içeriği tanıtıcı değeri
   uint&  width,         // piksel cinsinden genişlik
   uint&  height         // piksel cinsinden yükseklik
   );
```

Parametreler

context

[in] DXContextCreate()'te oluşturulan grafik içeriği için tanıtıcı değer.

width

[out] Piksel cinsinden kare genişliği.

height

[out] Piksel cinsinden kare yüksekliği.

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

DXContextClearColors

Render alma tamponu için tüm pikselleri belirtilen renge ayarlar.

```
bool  DXContextClearColors(
   int              context,      // grafik içeriği tanıtıcı değeri
   const DXVector&  color         // renk
   );
```

Parametreler

context

[in] DXContextCreate()'te oluşturulan grafik içeriği için tanıtıcı değer.

color

[in] Render rengi.

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

DXContextClearColors() fonksiyonu, bir sonraki kareyi render almadan önce renk tamponunu temizlemek için kullanılabilir.

DXContextClearDepth

Derinlik tamponunu temizler.

```
bool  DXContextClearDepth(
   int  context      // grafik içeriği tanıtıcı değeri
   );
```

Parametreler

context

[in] DXContextCreate()'te oluşturulan grafik içeriği için tanıtıcı değer.

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

DXContextClearDepth() fonksiyonu, bir sonraki kareyi render almadan önce derinlik tamponunu temizlemek için kullanılabilir.

DXContextGetColors

Grafik içeriğinden belirtilen boyutta ve ofsette bir görüntü elde eder.

```
bool  DXContextGetColors(
   int    context,                      // grafik içeriği tanıtıcı değeri
   uint&  image[],                      // görüntü piksel dizisi
   int    image_width=WHOLE_ARRAY,      // piksel cinsinden görüntü genişliği
   int    image_height=WHOLE_ARRAY,     // piksel cinsinden görüntü yüksekliği
   int    image_offset_x=0,             // X ofset
   int    image_offset_y=0              // Y ofset
   );
```

Parametreler

context

[in] DXContextCreate()'te oluşturulan grafik içeriği için tanıtıcı değer.

image

[out] ARGB formatında *image_width*image_height* piksel dizisi.

image_width=WHOLE_ARRAY

[in] Piksel cinsinden görüntü genişliği.

image_height=WHOLE_ARRAY

[in] Piksel cinsinden görüntü uzunluğu.

image_offset_x=0

[in] X ofset.

image_offset_y=0

[in] Y ofset.

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

DXContextGetDepth

Oluşturulan bir karenin derinlik tamponunu elde eder.

```
bool  DXContextGetDepth(
   int     context,      // grafik içeriği tanıtıcı değeri
   float&  image[]       // derinlik değeri dizisi
   );
```

Parametreler

context

[in] DXContextCreate()'te oluşturulan grafik içeriği için tanıtıcı değer.

image

[out] Render alınan karenin derinliğinin tampon değerleri dizisi.

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

Geri döndürülen tampon, DXContextGetColors()'da elde edilebilen render alınmış karenin her pikselinin derinliğini ilgili birimde (0.0'dan 1.0'a kadar) içerir.

DXBufferCreate

Veri dizisine dayalı olarak belirtilen tipte bir tampon oluşturur.

```
int  DXBufferCreate(
   int                  context,              // grafik içeriği tanıtıcı değeri
   ENUM_DX_BUFFER_TYPE  buffer_type,          // oluşturulan tampon tipi
   const void&          data[],               // tampon verisi
   uint                 start=0,              // başlangıç indeksi
   uint                 count=WHOLE_ARRAY     // eleman sayısı
   );
```

Parametreler

context

[in] DXContextCreate()'te oluşturulan grafik içeriği için tanıtıcı değer.

buffer_type

[in] ENUM_DX_BUFFER_TYPE listesinden tampon tipi.

data[]

[in] Tampon oluşturmak için veriler.

start

[in] Dizinin ilk elemanının indeksi. Bu noktadan itibaren dizinin verileri tampon için kullanılır. Varsayılan olarak, veriler dizinin başından itibaren alınır.

count

[in] Değer sayısı. Varsayılan olarak tüm dizi kullanılır (count=WHOLE_ARRAY).

Geri dönüş değeri

Oluşturulan tamponun tanıtıcı değeri ya da hata durumunda INVALID_HANDLE. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

İndeks tamponu için, *data[]* dizisi 'uint' tipinde olmalıdır; köşe tamponuna da köşeleri tanımlayan yapı dizisi aktarılır.

Artık kullanılmayan bir tanıtıcı değer, DXRelease() fonksiyonu tarafından açıkça serbest bırakılmalıdır.

ENUM_DX_BUFFER_TYPE

ID	Değer	Açıklama
DX_BUFFER_VERTEX	1	Köşe tamponu
DX_BUFFER_INDEX	2	İndeks tamponu

DXTextureCreate

Aktarılan görüntüden kesilmiş belirli bir boyuttaki dikdörtgenden 2D bir doku oluşturur.

```
int  DXTextureCreate(
   int              context,          // grafik içeriği tanıtıcı değeri
   ENUM_DX_FORMAT   format,           // piksel renk formatı
   uint             width,            // kaynak görüntü genişliği
   uint             height,           // kaynak görüntü yüksekliği
   const void&      data[],           // kaynak görüntü pikseli dizisi
   uint             data_x,           // doku oluşturmak için kullanılan dikdörtgenin X
   uint             data_y,           // doku oluşturmak için kullanılan dikdörtgenin Y
   uint             data_width,       // doku oluşturmak için kullanılan dikdörtgenin ge
   uint             data_height       // doku oluşturmak için kullanılan dikdörtgenin yü
   );
```

Parametreler

context

[in] DXContextCreate()'te oluşturulan grafik içeriği için tanıtıcı değer.

format

[in] ENUM_DX_FORMAT listesinden piksel renk formatı.

width

[in] Dokunun dayandığı görüntünün genişliği.

height

[in] Dokunun dayandığı görüntünün yüksekliği.

data

[in] Dokunun dayandığı görüntünün piksel dizisi.

data_x

[in] Doku oluşturmak için kullanılan dikdörtgenin X koordinatı (X ofset).

data_y

[in] Doku oluşturmak için kullanılan dikdörtgenin Y koordinatı (X ofset).

data_width

[in] Doku oluşturmak için kullanılan dikdörtgenin genişliği.

data_height

[in] Doku oluşturmak için kullanılan dikdörtgenin yüksekliği.

Geri dönüş değeri

Doku tanıtıcı değeri ya da hata durumunda INVALID_HANDLE. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

Artık kullanılmayan bir tanıtıcı değer, DXRelease() fonksiyonu tarafından açıkça serbest bırakılmalıdır.

ENUM_DX_FORMAT

ID	Değer	DXGI_FORMAT'taki eşi
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	Değer	DXGI_FORMAT'taki eşi
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	Değer	DXGI_FORMAT'taki eşi
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	Değer	DXGI_FORMAT'taki eşi
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	Değer	DXGI_FORMAT'taki eşi
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

Gölgelendirici girdilerini oluşturur.

```
int  DXInputCreate(
   int  context,          // grafik içeriği tanıtıcı değeri
   uint  input_size       // bayt cinsinden girdi boyutu
   );
```

Parametreler

context

[in] DXContextCreate()'te oluşturulan grafik içeriği için tanıtıcı değer.

input_size

[in] Parametre yapısının bayt cinsinden boyutu.

Geri dönüş değeri

Gölgelendirici girdileri için tanıtıcı değer ya da hata durumunda INVALID_HANDLE. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Artık kullanılmayan bir tanıtıcı değer, DXRelease() fonksiyonu tarafından açıkça serbest bırakılmalıdır.

DXInputSet

Gölgelendirici girdilerini ayarlar.

```
bool  DXInputSet(
   int           input,      // grafik içeriği tanıtıcı değeri
   const void&   data        // ayarlama için veriler
   );
```

Parametreler

input

[in] DXInputCreate()'te elde edilen gölgelendirici için girdi tanıtıcı değeri.

data

[in] Gölgelendirici girdilerini ayarlama verileri.

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

DXShaderCreate

Belirtilen tipte bir gölgelendirici oluşturur.

```
int  DXShaderCreate(
   int                  context,          // grafik içeriği tanıtıcı değeri
   ENUM_DX_SHADER_TYPE  shader_type,      // gölgelendirici tipi
   const string         source,           // gölgelendirici kaynak kodu
   const string         entry_point,      // giriş noktası
   string&              compile_error     // derleyici mesajlarını almak için dizge
   );
```

Parametreler

context

[in] DXContextCreate()'te oluşturulan grafik içeriği için tanıtıcı değer.

shader_type

[out] ENUM_DX_SHADER_TYPE listesinden bir değer geri döner.

source

[in] HLSL 5'teki gölgelendirici kaynak kodu.

entry_point

[in] Giriş noktası - kaynak kodundaki fonksiyon adı.

compile_error

[in] Derleme hatalarını almak için dizge.

Geri dönüş değeri

Gölgelendirici için tanıtıcı değer ya da hata durumunda INVALID_HANDLE. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

Artık kullanılmayan bir tanıtıcı değer, DXRelease() fonksiyonu tarafından açıkça serbest bırakılmalıdır.

ENUM_DX_SHADER_TYPE

ID	Değer	Açıklama
DX_SHADER_VERTEX	0	Köşe gölgelendirici
DX_SHADER_GEOMETRY	1	Geometri gölgelendirici
DX_SHADER_PIXEL	2	Piksel gölgelendirici

DXShaderSetLayout

Köşe gölgelendirici için köşe düzenini ayarlar.

```
bool  DXShaderSetLayout(
   int                     shader,       // gölgelendirici tanıtıcı değeri
   const DXVertexLayout&   layout[]      //
   );
```

Parametreler

shader

[in] DXShaderCreate()'te oluşturulan köşe gölgelendiricinin tanıtıcı değeri.

layout[]

[in] Köşe alanları açıklamalarının dizisi. Açıklamalar DXVertexLayout yapısı tarafından ayarlanır:

```
struct DXVertexLayout
  {
   string          semantic_name;       // Gölgelendirici girdi imzasında bu eleman
   uint            semantic_index;      // Elemanın semantik indeksi. Bir semantik
   ENUM_DX_FORMAT format;               // Eleman verilerinin veri tipi.
  };
```

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

Düzen, belirtilen köşe tamponundaki köşe tipiyle eşleşmelidir. Ayrıca, köşe gölgelendirici kodundaki giriş noktasında kullanılan köşe girdi tipleriyle de eşleşmelidir.

Gölgelendiricinin köşe tamponu DXBufferSet()'te ayarlanır.

DXVertexLayout yapısı, D3D11_INPUT_ELEMENT_DESC MSDN yapısının bir sürümüdür.

DXShaderInputsSet

Gölgelendirici girdilerini ayarlar.

```
bool  DXShaderInputsSet(
   int          shader,       // gölgelendirici tanıtıcı değeri
   const int&   inputs[]      // girdi tanıtıcı değerleri dizisi
   );
```

Parametreler

shader

[in] DXShaderCreate()'te oluşturulan gölgelendiricinin tanıtıcı değeri.

inputs[]

[in] DXInputCreate() kullanılarak oluşturulan girdi tanıtıcı değerlerinin dizisi.

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

Girdi parametresinin boyutu, gölgelendirici kodunda bildirilen cbuffer nesnesi sayısına eşit olmalıdır.

DXShaderTexturesSet

Gölgelendirici dokularını ayarlar.

```
bool  DXShaderTexturesSet(
   int          shader,          // gölgelendirici tanıtıcı değeri
   const  int&  textures[]       // yapı tanıtıcı değerleri dizisi
   );
```

Parametreler

shader

[in] DXShaderCreate()'te oluşturulan gölgelendiricinin tanıtıcı değeri.

textures[]

[in] DXTextureCreate() kullanılarak oluşturulan doku tanıtıcı değerlerinin dizisi.

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

Doku dizisinin boyutu, gölgelendirici kodunda bildirilen Texture2D nesnesi sayısına eşit olmalıdır.

DXDraw

DXBufferSet()'te ayarlanan köşe tamponunun köşe noktalarını render alır.

```
bool  DXDraw(
   int   context,               // grafik içeriği tanıtıcı değeri
   uint  start=0,               // ilk köşe indeksi
   uint  count=WHOLE_ARRAY      // köşe sayısı
   );
```

Parametreler

context

[in] DXContextCreate()'te oluşturulan grafik içeriği için tanıtıcı değer.

start

[in] Render alma için ilk köşenin indeksi.

count

[in] Render alınacak köşe sayısı.

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

Gölgelendiriciler, köşelerin renderı için DXShaderSet() kullanılarak önceden ayarlanmalıdır.

DXDrawIndexed

DXBufferSet() indeks tamponu tarafından tanımlanan grafik ilkellerini oluşturur.

```
bool  DXDrawIndexed(
   int   context,               // grafik içeriği tanıtıcı değeri
   uint  start=0,               // ilk ilkel indeksi
   uint  count=WHOLE_ARRAY      // ilkel sayısı
   );
```

Parametreler

context

[in] DXContextCreate()'te oluşturulan grafik içeriği için tanıtıcı değer.

start

[in] Render alma için ilk ilkelin indeksi.

count

[in] Render alınacak ilkel sayısı.

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

İndeks tamponu tarafından tanımlanan ilkellerin tipi, DXPrimiveTopologySet() kullanılarak ayarlanır.

DXBufferSet()'te köşe tamponu, ilkelleri render almak için önceden ayarlanmalıdır.

Ayrıca gölgelendiriciler önceden DXShaderSet() kullanılarak ayarlanmalıdır.

DXPrimiveTopologySet

DXDrawIndexed()'i kullanarak render almak için ilkellerin tipini ayarlar.

```
bool  DXPrimiveTopologySet(
   int                        context,              // grafik içeriği tanıtıcı değe
   ENUM_DX_PRIMITIVE_TOPOLOGY primitive_topology    // ilkel tip
   );
```

Parametreler

context

[in] DXContextCreate()'te oluşturulan grafik içeriği için tanıtıcı değer.

primitive_topology

[in] ENUM_DX_PRIMITIVE_TOPOLOGY listesinden bir değer geri döner.

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

ENUM_DX_PRIMITIVE_TOPOLOGY

ID	Değer	D3D11_PRIMITIVE_TOPOLOGY'deki eşi
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

Geçerli render işlemi için bir tampon ayarlar.

```
bool  DXBufferSet(
   int   context,              // grafik içeriği tanıtıcı değeri
   int   buffer,               // köşe veya indeks tamponu tanıtıcı değeri
   uint  start=0,              // başlangıç indeksi
   uint  count=WHOLE_ARRAY     // eleman sayısı
   );
```

Parametreler

context

[in] DXContextCreate()'te oluşturulan grafik içeriği için tanıtıcı değer.

buffer

[in] DXBufferCreate()'te oluşturulan köşe veya indeks tamponunun tanıtıcı değeri.

start

[in] Tamponun ilk elemanının indeksi. Varsayılan olarak, veriler tamponun başından itibaren alınır.

count

[in] Kullanılacak değer sayısı. Varsayılan olarak tüm tampon kullanılır.

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

DXDraw() kullanarak render alma adına köşe ve indeks tamponlarını ayarlamak için DXBufferSet() fonksiyonu çağrılmalıdır.

DXShaderSet

Render alma için gölgelendirici ayarlar.

```
bool  DXShaderSet(
   int  context,      // grafik içeriği tanıtıcı değeri
   int  shader        // gölgelendirici tanıtıcı değeri
   );
```

Parametreler

context

[in] DXContextCreate()'te oluşturulan grafik içeriği için tanıtıcı değer.

shader

[in] DXShaderCreate()'te oluşturulan gölgelendiricinin tanıtıcı değeri.

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

Birkaç gölgelendirici tipi aynı anda render işlemi için kullanılabilir (köşe, geometri ve piksel olanlar).

DXHandleType

Tanıtıcı değer tipini geri döndürür.

```
ENUM_DX_HANDLE_TYPE  DXHandleType(
   int  handle      // tanıtıcı değer
   );
```

Parametreler

handle

[in] Tanıtıcı değer.

Geri dönüş değeri

ENUM_DX_HANDLE_TYPE listesinden bir değer geri döner.

ENUM_DX_HANDLE_TYPE

ID	Değer	Açıklama
DX_HANDLE_INVALID	0	Geçersiz tanıtıcı değer
DX_HANDLE_CONTEXT	1	grafik içeriği tanıtıcı değeri
DX_HANDLE_SHADER	2	Gölgelendirici tanıtıcı değeri
DX_HANDLE_BUFFER	3	Köşe veya indeks tamponu tanıtıcı değeri
DX_HANDLE_INPUT	4	Gölgelendirici girdileri için tanıtıcı değer
DX_HANDLE_TEXTURE	5	Doku tanıtıcı değeri

DXRelease

Tanıtıcı değeri serbest bırakır.

```
bool  DXRelease(
   int  handle      // tanıtıcı değer
   );
```

Parametreler

context

[in] Serbest bırakılacak tanıtıcı değer.

Geri dönüş değeri

Eğer yürütme başarılı olursa true, aksi takdirde false olarak geri döner. Bir hata kodu almak için, GetLastError() fonksiyonu çağrılmalıdır.

Not

Artık kullanılmayan tüm tanıtıcı değerler, DXRelease() fonksiyonu tarafından açıkça serbest bırakılmalıdır.

Python ile entegrasyon için MetaTrader modülü

MQL5 finansal piyasalarda yüksek performanslı alım-satım uygulamalarının geliştirilmesi için tasarlanmıştır ve algoritmik alım-satımda kullanılan diğer özelleşmiş diller arasında benzersizdir. MQL5 programlarının sözdizimi ve hızı alabildiğince C++'a yakındır; OpenCL desteği ve MS Visual Studio ile entegrasyon vardır. İstatistik, bulanık mantık ve ALGLIB kütüphaneleri de mevcuttur. MetaEditor geliştirme ortamı, özel örtüler geliştirme gereksinimini ortadan kaldıran fonksiyonların "akıllı" bir şekilde içe aktarılması özelliğiyle .NET kitaplıkları için yerel destek sunar. Üçüncü parti C++ DLL'ler de kullanılabilir. C++ kaynak kodu dosyaları (CPP ve H) doğrudan editörden düzenlenebilir ve DLL halinde derlenebilir. Bunun için kullanıcının kişisel bilgisayarında kurulu olan Microsoft Visual Studio kullanılabilir.

Python, komut dosyaları ve uygulamalar geliştirmek için modern, üst düzey bir programlama dilidir. Makine öğrenmesi, süreç otomasyonu, veri analizi ve görselleştirme için birden çok kütüphane içerir.

Python için MetaTrader paketi, işlemciler arası iletişim yoluyla doğrudan MetaTrader 5 terminalinden hızlı ve rahat veri alışverişi için tasarlanmıştır. Bu şekilde alınan veriler, istatistiksel hesaplamalar ve makine öğrenmesi için de kullanılabilir.

Paketi komut satırından yükleme:

```
pip install MetaTrader5
```

Paketi komut satırından güncelleme:

```
pip install --upgrade MetaTrader5
```

MetaTrader 5 ve Python'u entegre etmek için fonksiyonlar

Fonksiyon	Eylem
initialize	MetaTrader 5 terminali ile bağlantı kurar
login	Belirtilen parametreleri kullanarak bir işlem hesabına bağlanır
shutdown	MetaTrader 5 terminaline önceden kurulmuş olan bağlantıyı kapatır
version	MetaTrader 5 terminal versiyonunu geri döndürür
last_error	Son hata ile ilgili verileri geri döndürür
account_info	Mevcut işlem hesabı ile ilgili bilgileri elde eder
terminal_Info	Bağlantı kurulmuş olan MetaTrader 5 terminalinin durumunu ve parametrelerini elde eder
symbols_total	MetaTrader 5 terminalindeki tüm finansal enstrümanların sayısını elde eder
symbols_get	MetaTrader 5 terminalinden tüm finansal enstrümanları elde eder
symbol_info	Belirtilen finansal enstrüman ile ilgili verileri elde eder
symbol_info_tick	Belirtilen finansal araç için son tiki elde eder
symbol_select	MarketWatch penceresinde bir sembol seçer veya pencereden bir sembol kaldırır

Fonksiyon	Eylem
market_book_add	MetaTrader 5 terminalini, belirli bir sembolün Piyasa Derinliği değişikliği olaylarına abone eder
market_book_get	Belirli bir sembolün Piyasa Derinliği girdilerini içeren veri grubunu BookInfo'dan geri döndürür
market_book_release	MetaTrader 5 terminalinin belirli bir sembolün Piyasa Derinliği değişikliği olaylarına aboneliğini iptal eder
copy_rates_from	MetaTrader 5 terminalinden belirtilen tarihten itibaren olan barları elde eder
copy_rates_from_pos	MetaTrader 5 terminalinden belirtilen indeksten itibaren olan barları elde eder
copyrates_range	MetaTrader 5 terminalinden belirtilen tarih aralığındaki barları elde eder
copy_ticks_from	MetaTrader 5 terminalinden belirtilen tarihten itibaren olan tikleri elde eder
copy_ticks_range	MetaTrader 5 terminalinden belirtilen tarih aralığındaki tikleri elde eder
orders_total	Aktif emirlerin sayısını elde eder
orders_get	Sembol veya etikete göre filtreleme özelliğiyle aktif emirleri elde eder
order_calc_margin	Belirtilen alım-satım işlemini gerçekleştirmek için hesap para birimi cinsinden marjini geri döndürür
order_calc_profit	Belirtilen alım-satım işlemi için hesap para birimi cinsinden kârı geri döndürür
order_check	Gerekli alım-satım işleminin gerçekleştirilmesi için fon yeterliliğini kontrol eder
order_send	Alım-satım işlemi gerçekleştirmek için bir istek gönderir
positions_total	Açık pozisyonların sayısını elde eder
positions_get	Sembol veya etikete göre filtreleme özelliğiyle açık pozisyonları elde eder
history_orders_total	İşlem geçmişinden belirtilen aralıktaki emir sayısını elde eder
history_orders_get	Pozisyon veya etikete göre filtreleme özelliğiyle işlem geçmişinden emirleri elde eder
history_deals_total	İşlem geçmişinden belirtilen aralıktaki işlem sayısını elde eder
history_deals_get	Pozisyon veya etikete göre filtreleme özelliğiyle işlem geçmişinden işlemleri elde eder

Python'u MetaTrader 5'e bağlamaya örnek

1. Python 3.8'in en son sürümünü şu adresten indirin: https://www.python.org/downloads/windows
2. Python'u kurarken, Python komut dosyalarını komut satırından çalıştırabilmek için "Add Python 3.8 to PATH%"ı işaretleyin.
3. MetaTrader 5 modülünü komut satırından yükleyin

```
pip install MetaTrader5
```

4. matplotlib ve pandas paketlerini ekleyin

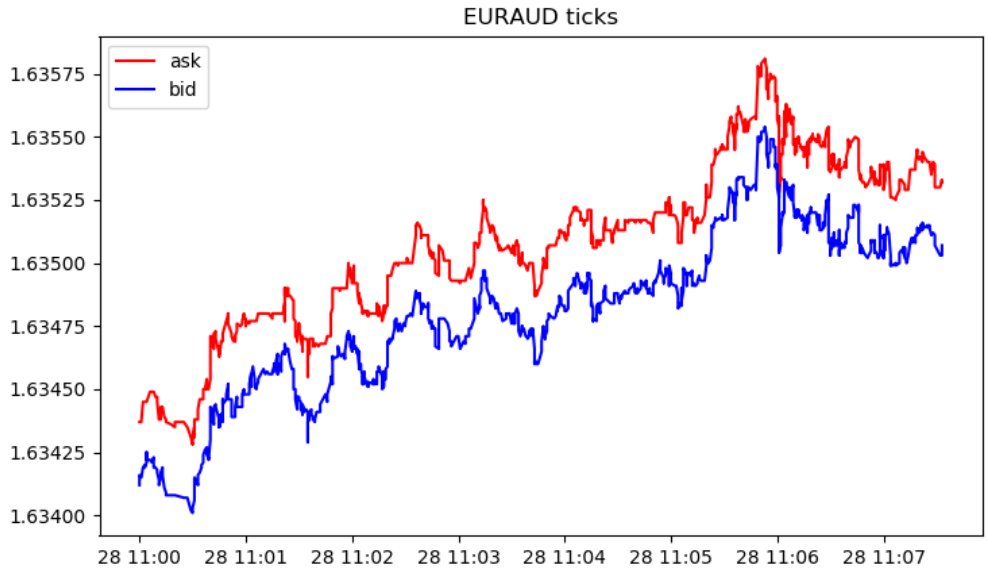
```
pip install matplotlib
pip install pandas
```

5. Test komut dosyasını başlatın

```
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

# MetaTrader 5'e bağlan
if not mt5.initialize():
    print("initialize() failed")
    mt5.shutdown()

# bağlantı durumunu ve parametreleri talep et
print(mt5.terminal_info())
# MetaTrader 5 sürümü ile ilgili verileri elde et
print(mt5.version())

# EURAUD'dan 1000 adet tik talep et
euraud_ticks = mt5.copy_ticks_from("EURAUD", datetime(2020,1,28,13), 1000, mt5.COPY_TI
# 2019.04.01 13:00 - 2019.04.02 13:00 aralığındaki AUDUSD tiklerini talep et
audusd_ticks = mt5.copy_ticks_range("AUDUSD", datetime(2020,1,27,13), datetime(2020,1,

# farklı sembollerden çeşitli şekillerde bar elde et
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)

# MetaTrader 5 terminaline olan bağlantıyı kapat
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
# elde edilen verilerden DataFrame oluştur
ticks_frame = pd.DataFrame(euraud_ticks)
# saniye cinsinden zamanı datetime biçimine dönüştür
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')
# grafik üzerinde tikleri görüntüle
plt.plot(ticks_frame['time'], ticks_frame['ask'], 'r-', label='ask')
plt.plot(ticks_frame['time'], ticks_frame['bid'], 'b-', label='bid')

# lejantı görüntüle
plt.legend(loc='upper left')

# başlık ekle
plt.title('EURAUD ticks')

# grafiği görüntüle
plt.show()
```

6. Verileri ve grafiği elde edin




```
[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

MetaTrader 5 terminali ile bağlantı kurar. Üç çağrı seçeneği vardır.

Parametresiz çağrı. Bağlantı için terminal otomatik olarak bulunur.

```
initialize()
```

Bağlanmanız gereken MetaTrader 5 terminalinin yolunu belirten çağrı.

```
initialize(
   path                       // MetaTrader 5 terminal EXE dosyasının yolu
   )
```

İşlem hesabının yolunu ve parametrelerini belirten çağrı.

```
initialize(
   path,                      // MetaTrader 5 terminal EXE dosyasının yolu
   login=LOGIN,               // hesap numarası
   password="PASSWORD",       // şifre
   server="SERVER",           // terminalde belirtildiği gibi sunucu adı
   timeout=TIMEOUT,           // zaman aşımı
   portable=False             // portable mod
   )
```

Parametreler

path

[in] Metatrader.exe veya metatrader64.exe dosyasının yolu. Opsiyonel adsız parametre. Parametre adı olmadan ilk olarak belirtilir. Yol belirtilmezse, modül yürütülebilir dosyayı kendi kendine bulmaya çalışır.

login=LOGIN

[in] İşlem hesabının numarası. Opsiyonel adlandırılmış parametre. Belirtilmezse, son işlem hesabı kullanılır.

password="PASSWORD"

[in] İşlem hesabının şifresi. Opsiyonel adlandırılmış parametre. Şifre belirtilmezse, belirtilen işlem hesabının terminal veritabanına kaydedilen şifresi otomatik olarak uygulanır.

server="SERVER"

[in] İşlem sunucusunun adı. Opsiyonel adlandırılmış parametre. Sunucu belirtilmezse, belirtilen işlem hesabının terminal veritabanına kaydedilen sunucusu otomatik olarak uygulanır.

timeout=TIMEOUT

[in] Milisaniye cinsinden bağlantı zaman aşımı. Opsiyonel adlandırılmış parametre. Belirtilmezse, 60 000 (60 saniye) değeri uygulanır.

portable=False

[in] Portable modda terminal başlatma bayrağı. Opsiyonel adlandırılmış parametre. Belirtilmezse, False değeri kullanılır.

Geri dönüş değeri

MetaTrader 5 terminaline başarılı bir şekilde bağlanılması durumunda True değeri geri döner, aksi takdirde False geri döner.

Not

Gerekirse, initialize() çağrısını yürütürken bağlantı kurmak için MetaTrader 5 terminali başlatılır.

Örnek:

```
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# belirtilen işlem hesabına MetaTrader 5 bağlantısı kur
if not mt5.initialize(login=25115284, server="MetaQuotes-Demo",password="4zatlbqx"):
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# bağlantı durumu, sunucu adı ve işlem hesabına ilişkin verileri görüntüle
print(mt5.terminal_info())
# MetaTrader 5 sürümü ile ilgili verileri görüntüle
print(mt5.version())

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()
```

Ayrıca bakınız

shutdown, terminal_info, version

login

Belirtilen parametreleri kullanarak bir işlem hesabına bağlanır.

```
login(
   login,                    // hesap numarası
   password="PASSWORD",      // şifre
   server="SERVER",          // terminalde belirtildiği gibi sunucu adı
   timeout=TIMEOUT           // zaman aşımı
   )
```

Parametreler

login

[in] İşlem hesabının numarası. Gerekli adsız parametre.

password

[in] İşlem hesabının şifresi. Opsiyonel adlandırılmış parametre. Şifre belirtilmezse, terminal veritabanına kaydedilen şifre otomatik olarak uygulanır.

server

[in] İşlem sunucusunun adı. Opsiyonel adlandırılmış parametre. Sunucu belirtilmezse, en son kullanılan sunucu otomatik olarak uygulanır.

timeout=TIMEOUT

[in] Milisaniye cinsinden bağlantı zaman aşımı. Opsiyonel adlandırılmış parametre. Belirtilmezse, 60 000 (60 saniye) değeri uygulanır. Belirtilen süre içerisinde bağlantı kurulmazsa, çağrı zorla sonlandırılır ve istisna oluşturulur.

Geri dönüş değeri

İşlem hesabına başarılı bir şekilde bağlanılması durumunda True, aksi takdirde False geri döner.

Örnek:

```
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# MetaTrader 5 sürümü ile ilgili verileri görüntüle
print(mt5.version())
# sunucu ve şifre belirtmeden alım-satım hesabına bağlan
account=17221085
authorized=mt5.login(account)  # bağlantı verileri hatırlanacak şekilde ayarlanmışsa t
if authorized:
    print("connected to account #{}".format(account))
else:
```

```
    print("failed to connect at account #{}, error code: {}".format(account, mt5.last_

# şifreyi belirterek başka bir işlem hesabına bağlan
account=25115284
authorized=mt5.login(account, password="gqrtz0lbdm")
if authorized:
    # elde edilen işlem hesabı verilerini 'olduğu gibi' görüntüle
    print(mt5.account_info())
    # işlem hesabı verilerini bir liste biçiminde görüntüle
    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("failed to connect at account #{}, error code: {}".format(account, mt5.last_

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()

Sonuç:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
[500, 2367, '23 Mar 2020']

connected to account #17221085

connected to account #25115284
AccountInfo(login=25115284, trade_mode=0, leverage=100, limit_orders=200, margin_so_mo
account properties:
   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.
```

Ayrıca bakınız

initialize, shutdown

shutdown

MetaTrader 5 terminaline önceden kurulmuş olan bağlantıyı kapatır.

```
shutdown()
```

Geri dönüş değeri

Yok.

Örnek:

```
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed")
    quit()

# bağlantı durumu, sunucu adı ve işlem hesabına ilişkin verileri görüntüle
print(mt5.terminal_info())
# MetaTrader 5 sürümü ile ilgili verileri görüntüle
print(mt5.version())

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()
```

Ayrıca bakınız

initialize, login_py, terminal_info, version

version

MetaTrader 5 terminal versiyonunu geri döndürür.

```
version()
```

Geri dönüş değeri

MetaTrader 5 terminal sürümü, yapı numarası ve sürüm tarihi geri döndürülür. Bir hata durumunda None geri döndürülür. Hata ile ilgili bilgiler last_error() kullanılarak elde edilebilir.

Not

version() fonksiyonu; terminal sürümünü, yapı numarasını ve sürüm tarihini üç değerden oluşan bir veri grubu halinde geri döndürür:

Tip	Açıklama	Örnek Değer
integer	MetaTrader 5 terminal versiyonu	500
integer	Yapı numarası	2007
string	Sürüm tarihi	'25 Feb 2019'

Örnek:

```
import MetaTrader5 as mt5
import pandas as pd
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# MetaTrader 5 sürümü ile ilgili verileri görüntüle
print(mt5.version())

# bağlantı durumu, sunucu adı ve işlem hesabı ile ilgili verileri 'olduğu gibi' görünt
print(mt5.terminal_info())
print()

# özellikleri sözlük biçiminde elde et
terminal_info_dict=mt5.terminal_info()._asdict()
# sözlüğü DataFrame'e dönüştür ve yazdır
df=pd.DataFrame(list(terminal_info_dict.items()),columns=['property','value'])
print("terminal_info() as dataframe:")
print(df[:-1])
```

```
# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()

Sonuç:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  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
```

Ayrıca bakınız

initialize, shutdown, terminal_info

last_error

Son hata ile ilgili verileri geri döndürür.

```
last_error()
```

Geri dönüş değeri

Son hata kodu ve açıklaması bir veri grubu halinde geri döndürülür.

Not

last_error(), MetaTrader 5 kütüphane fonksiyonunun başarısız bir şekilde yürütülmesi durumunda hata kodu alınmasına olanak sağlar. GetLastError() fonksiyonuna benzerdir. Ancak, kendi hata kodları uygulanır. Olası değerler:

Sabit	Değer	Açıklama
RES_S_OK	1	genel başarı
RES_E_FAIL	-1	genel hata
RES_E_INVALID_PARAMS	-2	geçersiz argümanlar/parametreler
RES_E_NO_MEMORY	-3	yetersiz bellek
RES_E_NOT_FOUND	-4	geçmiş yok
RES_E_INVALID_VERSION	-5	geçersiz sürüm
RES_E_AUTH_FAILED	-6	yetkilendirme başarısız
RES_E_UNSUPPORTED	-7	desteklenmeyen yöntem
RES_E_AUTO_TRADING_DISABLED	-8	otomatik alım-satım devre dışı
RES_E_INTERNAL_FAIL	-10000	dahili IPC, genel hata
RES_E_INTERNAL_FAIL_SEND	-10001	dahili IPC, gönderme başarısız
RES_E_INTERNAL_FAIL_RECEIVE	-10002	dahili IPC, elde etme başarısız
RES_E_INTERNAL_FAIL_INIT	-10003	dahili IPC, başlatma başarısız
RES_E_INTERNAL_FAIL_CONNECT	-10003	dahili IPC, ipc yok
RES_E_INTERNAL_FAIL_TIMEOUT	-10005	dahili zaman aşımı

Örnek:

```
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
```

```
    quit()

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()
```

Ayrıca bakınız

version, GetLastError

account_info

Mevcut işlem hesabı ile ilgili bilgileri elde eder.

```
account_info()
```

Geri dönüş değeri

Bilgiler adlandırılmış bir veri grubu yapısı (namedtuple) biçiminde geri döndürülür. Bir hata durumunda None geri döndürülür. Hata ile ilgili bilgiler last_error() kullanılarak elde edilebilir.

Not

Fonksiyon, tek bir çağrıda AccountInfoInteger, AccountInfoDouble ve AccountInfoString kullanılarak elde edilebilen tüm verileri geri döndürür.

Örnek:

```
import MetaTrader5 as mt5
import pandas as pd
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# sunucu ve şifreyi belirterek alım-satım hesabına bağlan
authorized=mt5.login(25115284, password="gqz0343lbdm")
if authorized:
    account_info=mt5.account_info()
    if account_info!=None:
        # elde edilen işlem hesabı verilerini 'olduğu gibi' görüntüle
        print(account_info)
        # işlem hesabı verilerini bir sözlük biçiminde görüntüle
        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()

       # sözlüğü DataFrame'e dönüştür ve yazdır
        df=pd.DataFrame(list(account_info_dict.items()),columns=['property','value'])
        print("account_info() as dataframe:")
        print(df)
else:
    print("failed to connect to trade account 25115284 with password=gqz0343lbdm, erro

# MetaTrader 5 terminaline olan bağlantıyı kapat
```

```
mt5.shutdown()


Sonuç:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  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.
```

Ayrıca bakınız

initialize, shutdown, login

terminal_info

Bağlı MetaTrader 5 müşteri terminalinin durumunu ve ayarlarını elde eder.

```
terminal_info()
```

Geri dönüş değeri

Bilgiler adlandırılmış bir veri grubu yapısı (namedtuple) biçiminde geri döndürülür. Bir hata durumunda None geri döndürülür. Hata ile ilgili bilgiler last_error() kullanılarak elde edilebilir.

Not

Fonksiyon, tek bir çağrıda TerminalInfoInteger, TerminalInfoDouble ve TerminalInfoDouble kullanılarak elde edilebilen tüm verileri geri döndürür.

Örnek:

```
import MetaTrader5 as mt5
import pandas as pd
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# MetaTrader 5 sürümü ile ilgili verileri görüntüle
print(mt5.version())
# terminal ayarları ve durumu ile ilgili bilgileri görüntüle
terminal_info=mt5.terminal_info()
if terminal_info!=None:
    # terminal verilerini 'olduğu gibi' görüntüle
    print(terminal_info)
    # verileri bir liste biçiminde görüntüle
    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()
   # sözlüğü DataFrame'e dönüştür ve yazdır
    df=pd.DataFrame(list(terminal_info_dict.items()),columns=['property','value'])
    print("terminal_info() as dataframe:")
    print(df)

# MetaTrader 5 terminaline olan bağlantıyı kapat
```

```
mt5.shutdown()


Sonuç:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  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
```

Ayrıca bakınız

initialize, shutdown, version

symbols_total

MetaTrader 5 terminalindeki tüm finansal enstrümanların sayısını elde eder.

```
symbols_total()
```

Geri dönüş değeri

Tam sayı değeri.

Not

Fonksiyon, SymbolsTotal()'e benzerdir. Ancak, kullanıcı tanımlı olanlar ve MarketWatch'ta devre dışı bırakılanlar dahil tüm sembollerin sayısını geri döndürür.

Örnek:

```
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# finansal enstrüman sayısını elde et
symbols=mt5.symbols_total()
if symbols>0:
    print("Total symbols =",symbols)
else:
    print("symbols not found")

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()
```

Ayrıca bakınız

symbols_get, symbol_select, symbol_info

symbols_get

MetaTrader 5 terminalinden tüm finansal enstrümanları elde eder.

```
symbols_get(
   group="GROUP"      // sembol filtresi
)
```

group="GROUP"

[in] Gerekli sembollerden oluşan bir grup düzenlemek için filtre. Opsiyonel parametre. Bir grup belirtilirse, fonksiyon yalnızca belirtilen ölçütleri karşılayan sembolleri geri döndürür.

Geri dönüş değeri

Semboller bir veri grubu şeklinde geri döndürülür. Bir hata durumunda None geri döndürülür. Hata ile ilgili bilgiler last_error() kullanılarak elde edilebilir.

Not

group parametresi, sembolleri ada göre filtrelemenize olanak tanır. '*' bir dizgenin başında ve sonunda kullanılabilir.

group parametresi, adlandırılmış veya adsız olarak kullanılabilir. Her iki seçenek de aynı şekilde çalışır. Adlandırılmış seçenek (group="GROUP") kodun okunmasını kolaylaştırır.

group parametresi, virgülle ayrılmış birkaç koşul içerebilir. Bir koşul '*' kullanılarak maske olarak ayarlanabilir. Hariç tutma için '!' mantıksal olumsuzlama sembolü kullanılabilir. Tüm koşullar sırayla uygulanır, yani önce gruba dahil etme koşullarını ve ardından dışlama koşulunu belirtmelisiniz. Örneğin, *group*="*, !EUR", önce tüm sembollerin seçilmesi ve daha sonra adlarında "EUR" içerenlerin hariç tutulması gerektiği anlamına gelir.

symbol_info() fonksiyonundan farklı olarak, symbols_get() fonksiyonu, tek bir çağrıda istenen tüm sembollerle ilgili verileri geri döndürür.

Örnek:

```
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# tüm sembolleri elde et
symbols=mt5.symbols_get()
print('Symbols: ', len(symbols))
count=0
# ilk beşini görüntüle
for s in symbols:
    count+=1
```

```
    print("{}. {}".format(count,s.name))
    if count==5: break
print()

# adlarında RU içeren sembolleri elde et
ru_symbols=mt5.symbols_get("*RU*")
print('len(*RU*): ', len(ru_symbols))
for s in ru_symbols:
    print(s.name)
print()

# adlarında USD, EUR, JPY ve GBP içermeyen sembolleri elde et
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)

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()

Sonuç:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  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_d
AUDNZD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_d
CADCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_d
NZDCAD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_d
NZDCHF : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_d
NZDSGD : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_d
CADMXN : SymbolInfo(custom=False, chart_mode=0, select=False, visible=False, session_d
```

```
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
```

Ayrıca bakınız

symbols_total, symbol_select, symbol_info

symbol_info

Belirtilen finansal enstrüman ile ilgili verileri elde eder.

```
symbol_info(
   symbol      // finansal enstrüman adı
)
```

symbol

[in] Finansal enstrüman adı. Gerekli adsız parametre.

Geri dönüş değeri

Bilgiler adlandırılmış bir veri grubu yapısı (namedtuple) biçiminde geri döndürülür. Bir hata durumunda None geri döndürülür. Hata ile ilgili bilgiler last_error() kullanılarak elde edilebilir.

Not

Fonksiyon, tek bir çağrıda SymbolInfoInteger, SymbolInfoDouble ve SymbolInfoString kullanılarak elde edilebilen tüm verileri geri döndürür.

Örnek:

```
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# MarketWatch'ta EURJPY sembolünün görüntülenmesini etkinleştirmeyi dene
selected=mt5.symbol_select("EURJPY",True)
if not selected:
    print("Failed to select EURJPY")
    mt5.shutdown()
    quit()

# EURJPY sembol özelliklerini görüntüle
symbol_info=mt5.symbol_info("EURJPY")
if symbol_info!=None:
    # terminal verilerini 'olduğu gibi' görüntüle
    print(symbol_info)
    print("EURJPY: spread =",symbol_info.spread,"  digits =",symbol_info.digits)
    # sembol özelliklerini bir liste halinde görüntüle
    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]))
```

```
# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()


Sonuç:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  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
```

Ayrıca bakınız

account_info, terminal_info

symbol_info_tick

Belirtilen finansal araç için son tiki elde eder.

```
symbol_info_tick(
   symbol      // finansal enstrüman adı
)
```

symbol

[in] Finansal enstrüman adı. Gerekli adsız parametre.

Geri dönüş değeri

Bilgiler bir veri grubu şeklinde geri döndürülür. Bir hata durumunda None geri döndürülür. Hata ile ilgili bilgiler last_error() kullanılarak elde edilebilir.

Not

Fonksiyon, SymbolInfoTick'e benzerdir.

Örnek:

```
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# MarketWatch'ta GBPUSD sembolünün görüntülenmesini etkinleştirmeyi dene
selected=mt5.symbol_select("GBPUSD",True)
if not selected:
    print("Failed to select GBPUSD")
    mt5.shutdown()
    quit()

# son GBPUSD tikini görüntüle
lasttick=mt5.symbol_info_tick("GBPUSD")
print(lasttick)
# tik alanı değerlerini liste biçiminde görüntüle
print("Show 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]))

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()
```

```
Sonuç:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  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
```

Ayrıca bakınız

symbol_info, symbol_info

symbol_select

MarketWatch penceresinde bir sembol seçer veya pencereden bir sembol kaldırır.

```
symbol_select(
   symbol,      // finansal enstrüman adı
   enable=None  // etkinleştir veya devre dışı bırak
)
```

symbol

[in] Finansal enstrüman adı. Gerekli adsız parametre.

enable

[in] Seçim. Opsiyonel adsız parametre. 'false' ise, sembol MarketWatch penceresinden kaldırılır. Aksi takdirde, sembol MarketWatch penceresinde seçilir. Sembole ait açık grafikler varsa veya sembol üzerinde pozisyonlar açılmışsa, sembol kaldırılamaz.

Geri dönüş değeri

Başarılıysa - true, aksi takdirde - false.

Not

Fonksiyon, SymbolSelect'e benzerdir.

Örnek:

```
import MetaTrader5 as mt5
import pandas as pd
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize(login=25115284, server="MetaQuotes-Demo",password="4zatlbqx"):
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# MarketWatch'ta EURCAD sembolünün görüntülenmesini etkinleştirmeyi dene
selected=mt5.symbol_select("EURCAD",True)
if not selected:
    print("Failed to select 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()

    # sembol özelliklerini bir sözlük biçiminde görüntüle
    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()

   # sözlüğü DataFrame'e dönüştür ve yazdır
    df=pd.DataFrame(list(symbol_info_dict.items()),columns=['property','value'])
    print("symbol_info_dict() as dataframe:")
    print(df)

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()


Sonuç:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  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]
```

Ayrıca bakınız

[symbol_info](#)

market_book_add

MetaTrader 5 terminalini, belirli bir sembolün Piyasa Derinliği değişikliği olaylarına abone eder.

```
market_book_add(
   sembol      // finansal enstrüman adı
)
```

symbol

Finansal enstrüman adı. Gerekli adsız parametre.

Geri dönüş değeri

Başarılıysa - true, aksi takdirde - false.

Not

Fonksiyon, MarketBookAdd'e benzerdir.

Ayrıca bakınız

market_book_get, market_book_release, Piyasa Derinliği yapısı

market_book_get

Belirli bir sembolün Piyasa Derinliği girdilerini içeren veri grubunu BookInfo'dan geri döndürür.

```
market_book_get(
   sembol      // finansal enstrüman adı
)
```

symbol

Finansal enstrüman adı. Gerekli adsız parametre.

Geri dönüş değeri

Piyasa Derinliği içeriğini BookInfo girdilerinden bir veri grubu (emir tipi, fiyat ve lot cinsinden hacim) halinde geri döndürür. BookInfo, MqlBookInfo yapısına benzerdir.

Bir hata durumunda None geri döndürülür. Hata ile ilgili bilgiler last_error() kullanılarak elde edilebilir.

Not

Piyasa Derinliği değişikliği olaylarına abonelik, önceden market_book_add() fonksiyonu kullanılarak gerçekleştirilmelidir.

Fonksiyon, MarketBookGet'e benzerdir.

Örnek:

```
import MetaTrader5 as mt5
import time
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print("")

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
   # MetaTrader 5 terminaline olan bağlantıyı kapat
    mt5.shutdown()
    quit()

# EURUSD için piyasa derinliği (Depth of Market) güncellemelerine abone ol
if mt5.market_book_add('EURUSD'):
  # piyasa derinliği verilerini bir döngüde 10 kez al
   for i in range(10):
        # pazar derinliği (Depth of Market) içeriğini al
        items = mt5.market_book_get('EURUSD')
        # tüm piyasa derinliğini tek bir dizgede 'olduğu gibi' görüntüle
        print(items)
        # şimdi daha fazla netlik için her bir emri ayrı ayrı göster
```

```
        if items:
            for it in items:
                # emir içeriği
                print(it._asdict())
        # piyasa derinliği verilerinin bir sonraki talebinden önce 5 saniye durakla
        time.sleep(5)
  # piyasa derinliği (Depth of Market) güncellemelerine aboneliği iptal et
   mt5.market_book_release('EURUSD')
else:
    print("mt5.market_book_add('EURUSD') failed, error code =",mt5.last_error())

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()


Sonuç:
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}
```

Ayrıca bakınız

market_book_add, market_book_release, Piyasa Derinliği yapısı

market_book_release

MetaTrader 5 terminalinin belirli bir sembolün Piyasa Derinliği değişikliği olaylarına aboneliğini iptal eder.

```
market_book_release(
   sembol      // finansal enstrüman adı
)
```

symbol

Finansal enstrüman adı. Gerekli adsız parametre.

Geri dönüş değeri

Başarılıysa - true, aksi takdirde - false.

Not

Fonksiyon, MarketBookRelease'e benzerdir.

Ayrıca bakınız

market_book_add, market_book_get, Piyasa Derinliği yapısı

copy_rates_from

MetaTrader 5 terminalinden belirtilen tarihten itibaren olan barları elde eder.

```
copy_rates_from(
   symbol,       // sembol adı
   timeframe,    // zaman aralığı
   date_from,    // başlangıcın gerçekleşeceği bar açılış tarihi
   count         // bar sayısı
   )
```

Parametreler

symbol

[in] Finansal enstrüman adı, örneğin, "EURUSD". Gerekli adsız parametre.

timeframe

[in] Barların talep edileceği zaman aralığı. TIMEFRAME sayımından bir değer ayarlanır. Gerekli adsız parametre.

date_from

[in] İstenilen seçimin ilk barının açılış tarihi. 'datetime' nesnesi tarafından veya 1970.01.01'den beri geçen saniye sayısı olarak ayarlanır. Gerekli adsız parametre.

count

[in] Elde edilecek bar sayısı. Gerekli adsız parametre.

Geri dönüş değeri

Barlar time, open, high, low, close, tick_volume, spread ve real_volume adlı sütunlara sahip numpy dizisi halinde geri döndürülür. Bir hata durumunda None geri döndürülür. Hata ile ilgili bilgiler last_error() kullanılarak elde edilebilir.

Not

Daha fazla bilgi için CopyRates() fonksiyonuna bakın.

Sadece belirtilen tarihe eşit veya ondan daha önceki verilerin dönüşü yapılacaktır. Yani, herhangi bir çubuğun açılış zamanı her zaman belirtilenden az veya belirtilene eşit olacaktır.

MetaTrader 5 terminali yalnızca grafiklerde kullanıcıya kullanılabilir olan geçmişteki barları sağlar. Kullanıcılar için kullanılabilir bar sayısı "Maks. bar sayısı" parametresinde ayarlanır.

Python, 'datetime' nesnesini oluştururken yerel zaman dilimini kullanır, MetaTrader 5 ise tik ve bar açılış zamanını UTC zaman diliminde (öteleme olmadan) depolar. Bu nedenle; 'datetime', zamanı kullanan fonksiyonları yürütmek için UTC zamanında oluşturulmalıdır. MetaTrader 5 terminalinden alınan veriler UTC zamanına sahiptir.

TIMEFRAME, olası grafik periyodu değerlerine sahip bir sayımdır

ID	Açıklama
TIMEFRAME_M1	1 dakika

ID	Açıklama
TIMEFRAME_M2	2 dakika
TIMEFRAME_M3	3 dakika
TIMEFRAME_M4	4 dakika
TIMEFRAME_M5	5 dakika
TIMEFRAME_M6	6 dakika
TIMEFRAME_M10	10 dakika
TIMEFRAME_M12	12 dakika
TIMEFRAME_M12	15 dakika
TIMEFRAME_M20	20 dakika
TIMEFRAME_M30	30 dakika
TIMEFRAME_H1	1 saat
TIMEFRAME_H2	2 saat
TIMEFRAME_H3	3 saat
TIMEFRAME_H4	4 saat
TIMEFRAME_H6	6 saat
TIMEFRAME_H8	8 saat
TIMEFRAME_H12	12 saat
TIMEFRAME_D1	1 gün
TIMEFRAME_W1	1 hafta
TIMEFRAME_MN1	1 ay

Örnek:

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# elde edilen verileri tablo şeklinde görüntülemek için 'pandas' modülünü içe aktar
import pandas as pd
pd.set_option('display.max_columns', 500) # görüntülenecek sütun sayısı
pd.set_option('display.width', 1500)      # görüntülenecek maksimum tablo genişliği
# zaman dilimi ile çalışmak için pytz modülünü içe aktar
import pytz
```

```
# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# zaman dilimini UTC olarak ayarla
timezone = pytz.timezone("Etc/UTC")
# yerel zaman dilimi ötelemesinin gerçekleşmesini önlemek için UTC zaman diliminde 'da
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
# UTC zaman diliminde 01.10.2020 tarihinden itibaren 10 EURUSD H4 barı elde et
rates = mt5.copy_rates_from("EURUSD", mt5.TIMEFRAME_H4, utc_from, 10)

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()
# elde edilen verilerin her bir elemanını yeni bir satırda göster
print("Elde edilen verileri 'olduğu gibi' görüntüle")
for rate in rates:
    print(rate)

# elde edilen verilerden DataFrame oluştur
rates_frame = pd.DataFrame(rates)
# saniye cinsinden zamanı datetime biçimine dönüştür
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# verileri görüntüle
print("\nVerileri içeren veri çerçevesini görüntüle")
print(rates_frame)
```

```
Sonuç:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Elde edilen verileri 'olduğu gibi' görüntüle
(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)

Verileri içeren veri çerçevesini görüntüle
                 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

Ayrıca bakınız

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

copy_rates_from_pos

MetaTrader 5 terminalinden belirtilen indeksten itibaren olan barları elde eder.

```
copy_rates_from_pos(
   symbol,       // sembol adı
   timeframe,    // zaman aralığı
   start_pos,    // başlangıç barının indeksi
   count         // bar sayısı
   )
```

Parametreler

symbol

[in] Finansal enstrüman adı, örneğin, "EURUSD". Gerekli adsız parametre.

timeframe

[in] Barların talep edileceği zaman aralığı. TIMEFRAME sayımından bir değer ayarlanır. Gerekli adsız parametre.

start_pos

[in] Verilerin talep edilmeye başlanacağı barın indeksi. Barların numaralandırılması günümüzden geçmişi doğru gider. Böylece, sıfır barı mevcut barı ifade eder. Gerekli adsız parametre.

count

[in] Elde edilecek bar sayısı. Gerekli adsız parametre.

Geri dönüş değeri

Barlar time, open, high, low, close, tick_volume, spread ve real_volume adlı sütunlara sahip numpy dizisi halinde geri döndürülür. Bir hata durumunda None geri döndürülür. Hata ile ilgili bilgiler last_error() kullanılarak elde edilebilir.

Not

Daha fazla bilgi için CopyRates() fonksiyonuna bakın.

MetaTrader 5 terminali yalnızca grafiklerde kullanıcıya kullanılabilir olan geçmişteki barları sağlar. Kullanıcılar için kullanılabilir bar sayısı "Maks. bar sayısı" parametresinde ayarlanır.

Örnek:

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# elde edilen verileri tablo şeklinde görüntülemek için 'pandas' modülünü içe aktar
import pandas as pd
pd.set_option('display.max_columns', 500) # görüntülenecek sütun sayısı
pd.set_option('display.width', 1500)      # görüntülenecek maksimum tablo genişliği
```

```
# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# Geçerli günden itibaren 10 GBPUSD D1 barı elde et
rates = mt5.copy_rates_from_pos("GBPUSD", mt5.TIMEFRAME_D1, 0, 10)

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()
# elde edilen verilerin her bir elemanını yeni bir satırda göster
print("Elde edilen verileri 'olduğu gibi' görüntüle")
for rate in rates:
    print(rate)

# elde edilen verilerden DataFrame oluştur
rates_frame = pd.DataFrame(rates)
# saniye cinsinden zamanı datetime biçimine dönüştür
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# verileri görüntüle
print("\nVerileri içeren veri çerçevesini görüntüle")
print(rates_frame)
```

Sonuç:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Elde edilen verileri 'olduğu gibi' görüntüle
(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)

Verileri içeren veri çerçevesini görüntüle
        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

Ayrıca bakınız

CopyRates, copy_rates_from, copy_rates_range, copy_ticks_from, copy_ticks_range

copy_rates_range

MetaTrader 5 terminalinden belirtilen tarih aralığındaki barları elde eder.

```
copy_rates_range(
   symbol,       // sembol adı
   timeframe,    // zaman aralığı
   date_from,    // barların talep edileceği aralığın başlangıç tarihi
   date_to       // barların talep edileceği aralığın bitiş tarihi
   )
```

Parametreler

symbol

[in] Finansal enstrüman adı, örneğin, "EURUSD". Gerekli adsız parametre.

timeframe

[in] Barların talep edileceği zaman aralığı. TIMEFRAME sayımından bir değer ayarlanır. Gerekli adsız parametre.

date_from

[in] Barların talep edileceği aralığın başlangıç tarihi. 'datetime' nesnesi tarafından veya 1970.01.01'den beri geçen saniye sayısı olarak ayarlanır. Açılış zamanı >= date_from olan barlar geri döndürülür. Gerekli adsız parametre.

date_to

[in] Barların talep edileceği aralığın bitiş tarihi. 'datetime' nesnesi tarafından veya 1970.01.01'den beri geçen saniye sayısı olarak ayarlanır. Açılış zamanı <= date_to olan barlar geri döndürülür. Gerekli adsız parametre.

Geri dönüş değeri

Barlar time, open, high, low, close, tick_volume, spread ve real_volume adlı sütunlara sahip numpy dizisi halinde geri döndürülür. Bir hata durumunda None geri döndürülür. Hata ile ilgili bilgiler last_error() kullanılarak elde edilebilir.

Not

Daha fazla bilgi için CopyRates() fonksiyonuna bakın.

MetaTrader 5 terminali yalnızca grafiklerde kullanıcıya kullanılabilir olan geçmişteki barları sağlar. Kullanıcılar için kullanılabilir bar sayısı "Maks. bar sayısı" parametresinde ayarlanır.

Python, 'datetime' nesnesini oluştururken yerel zaman dilimini kullanır, MetaTrader 5 ise tik ve bar açılış zamanını UTC zaman diliminde (öteleme olmadan) depolar. Bu nedenle; 'datetime', zamanı kullanan fonksiyonları yürütmek için UTC zamanında oluşturulmalıdır. MetaTrader 5 terminalinden alınan veriler UTC zamanına sahiptir.

Örnek:

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
```

```
print("MetaTrader5 package version: ",mt5.__version__)

# elde edilen verileri tablo şeklinde görüntülemek için 'pandas' modülünü içe aktar
import pandas as pd
pd.set_option('display.max_columns', 500) # görüntülenecek sütun sayısı
pd.set_option('display.width', 1500)      # görüntülenecek maksimum tablo genişliği
# zaman dilimi ile çalışmak için pytz modülünü içe aktar
import pytz

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# zaman dilimini UTC olarak ayarla
timezone = pytz.timezone("Etc/UTC")
# yerel zaman dilimi ötelemesinin gerçekleşmesini önlemek için UTC zaman diliminde 'da
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, hour = 13, tzinfo=timezone)
# UTC zaman diliminde 2020.01.10 00:00 - 2020.01.11 13:00 aralığında USDJPY M5 barları
rates = mt5.copy_rates_range("USDJPY", mt5.TIMEFRAME_M5, utc_from, utc_to)

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()

# elde edilen verilerin her bir elemanını yeni bir satırda göster
print("Elde edilen verileri 'olduğu gibi' görüntüle")
counter=0
for rate in rates:
    counter+=1
    if counter<=10:
        print(rate)

# elde edilen verilerden DataFrame oluştur
rates_frame = pd.DataFrame(rates)
# saniye cinsinden zamanı 'datetime' biçimine dönüştür
rates_frame['time']=pd.to_datetime(rates_frame['time'], unit='s')

# verileri görüntüle
print("\nVerileri içeren veri çerçevesini görüntüle")
print(rates_frame.head(10))
```

Sonuç:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Elde edilen verileri 'olduğu gibi' görüntüle
(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)

Verileri içeren veri çerçevesini görüntüle
                 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
```

Ayrıca bakınız

CopyRates, copy_rates_from, copy_rates_range, copy_ticks_from, copy_ticks_range

copy_ticks_from

MetaTrader 5 terminalinden belirtilen tarihten itibaren olan tikleri elde eder.

```
copy_ticks_from(
   symbol,       // sembol adı
   date_from,    // tiklerin talep edilmesinin başlanacağı tarih
   count,        // talep edilen tik sayısı
   flags         // talep edilen tiklerin tipini tanımlayan bayrakların kombinasyonu
   )
```

Parametreler

symbol

[in] Finansal enstrüman adı, örneğin, "EURUSD". Gerekli adsız parametre.

from

[in] Tiklerin talep edilmesinin başlanacağı tarih. 'datetime' nesnesi tarafından veya 1970.01.01'den beri geçen saniye sayısı olarak ayarlanır. Gerekli adsız parametre.

count

[in] Elde edilecek tik sayısı. Gerekli adsız parametre.

flags

[in] Talep edilen tiklerin tipini tanımlayan bayrak. COPY_TICKS_INFO - Alış ve/veya Satış fiyatı değişimleri sonucu oluşan tikler, COPY_TICKS_TRADE - Son ve/veya Hacim değişimleri sonucu oluşan tikler, COPY_TICKS_ALL - tüm tikler. Bayrak değerleri COPY_TICKS sayımında açıklanmıştır. Gerekli adsız parametre.

Geri dönüş değeri

Tikler time, bid, ask, last ve flags adlı sütunlara sahip numpy dizisi halinde geri döndürülür. 'flags' değeri, TICK_FLAG sayımındaki bayrakların bir kombinasyonu olabilir. Bir hata durumunda None geri döndürülür. Hata ile ilgili bilgiler last_error() kullanılarak elde edilebilir.

Not

Daha fazla bilgi için CopyTicks fonksiyonuna bakın.

Python, 'datetime' nesnesini oluştururken yerel zaman dilimini kullanır, MetaTrader 5 ise tik ve bar açılış zamanını UTC zaman diliminde (öteleme olmadan) depolar. Bu nedenle; 'datetime', zamanı kullanan fonksiyonları yürütmek için UTC zamanında oluşturulmalıdır. MetaTrader 5 terminalinden alınan veriler UTC zamanına sahiptir.

COPY_TICKS; copy_ticks_from() ve copy_ticks_range() fonksiyonları kullanılarak talep edilebilecek tiklerin tiplerini tanımlar.

ID	Açıklama
COPY_TICKS_ALL	tüm tikler
COPY_TICKS_INFO	Alış ve/veya Satış fiyatı değişimleri sonucu oluşan tikler

ID	Açıklama
COPY_TICKS_TRADE	Son ve/veya Hacim değişimleri sonucu oluşan tikler

TICK_FLAG, tikler için olası bayrakları tanımlar. Bu bayraklar, copy_ticks_from() ve copy_ticks_range() fonksiyonları tarafından elde edilen tikleri tanımlamak için kullanılır.

ID	Açıklama
TICK_FLAG_BID	Alış (Bid) fiyatı değişti
TICK_FLAG_ASK	Satış (Ask) fiyatı değişti
TICK_FLAG_LAST	Son fiyat değişti
TICK_FLAG_VOLUME	Hacim değişti
TICK_FLAG_BUY	son Alış (Buy) fiyatı değişti
TICK_FLAG_SELL	son Satış (Sell) fiyatı değişti

Örnek:

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# elde edilen verileri tablo şeklinde görüntülemek için 'pandas' modülünü içe aktar
import pandas as pd
pd.set_option('display.max_columns', 500) # görüntülenecek sütun sayısı
pd.set_option('display.width', 1500)      # görüntülenecek maksimum tablo genişliği
# zaman dilimi ile çalışmak için pytz modülünü içe aktar
import pytz

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# zaman dilimini UTC olarak ayarla
timezone = pytz.timezone("Etc/UTC")
# yerel zaman dilimi ötelemesinin gerçekleşmesini önlemek için UTC zaman diliminde 'da
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
# UTC zaman diliminde 10.01.2019 tarihinden başlayarak 100.000 EURUSD tiki talep et
ticks = mt5.copy_ticks_from("EURUSD", utc_from, 100000, mt5.COPY_TICKS_ALL)
print("Elde edilen tik sayısı:",len(ticks))

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()
```

```
# her bir tikle ilgili verileri yeni bir satırda göster
print("Elde edilen tikleri 'olduğu gibi' görüntüle")
count = 0
for tick in ticks:
    count+=1
    print(tick)
    if count >= 10:
        break

# elde edilen verilerden DataFrame oluştur
ticks_frame = pd.DataFrame(ticks)
# saniye cinsinden zamanı datetime biçimine dönüştür
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')

# verileri görüntüle
print("\nTikleri içeren veri çerçevesini görüntüle")
print(ticks_frame.head(10))

Sonuç:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Elde edilen tik sayısı: 100000
Elde edilen tikleri 'olduğu gibi' görüntüle
(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.)

Tikleri içeren veri çerçevesini görüntüle
                  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
```

Ayrıca bakınız

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

copy_ticks_range

MetaTrader 5 terminalinden belirtilen tarih aralığındaki tikleri elde eder.

```
copy_ticks_range(
   symbol,       // sembol adı
   date_from,    // tiklerin talep edileceği aralığın başlangıç tarihi
   date_to,      // tiklerin talep edileceği aralığın bitiş tarihi
   flags         // talep edilen tiklerin tipini tanımlayan bayrakların kombinasyonu
   )
```

Parametreler

symbol

[in] Finansal enstrüman adı, örneğin, "EURUSD". Gerekli adsız parametre.

date_from

[in] Tiklerin talep edilmesinin başlanacağı tarih. 'datetime' nesnesi tarafından veya 1970.01.01'den beri geçen saniye sayısı olarak ayarlanır. Gerekli adsız parametre.

date_to

[in] Tiklerin talep edileceği aralığın bitiş tarihi. 'datetime' nesnesi tarafından veya 1970.01.01'den beri geçen saniye sayısı olarak ayarlanır. Gerekli adsız parametre.

flags

[in] Talep edilen tiklerin tipini tanımlayan bayrak. COPY_TICKS_INFO - Alış ve/veya Satış fiyatı değişimleri sonucu oluşan tikler, COPY_TICKS_TRADE - Son ve/veya Hacim değişimleri sonucu oluşan tikler, COPY_TICKS_ALL - tüm tikler. Bayrak değerleri COPY_TICKS sayımında açıklanmıştır. Gerekli adsız parametre.

Geri dönüş değeri

Tikler time, bid, ask, last ve flags adlı sütunlara sahip numpy dizisi halinde geri döndürülür. 'flags' değeri, TICK_FLAG sayımındaki bayrakların bir kombinasyonu olabilir. Bir hata durumunda None geri döndürülür. Hata ile ilgili bilgiler last_error() kullanılarak elde edilebilir.

Not

Daha fazla bilgi için CopyTicks fonksiyonuna bakın.

Python, 'datetime' nesnesini oluştururken yerel zaman dilimini kullanır, MetaTrader 5 ise tik ve bar açılış zamanını UTC zaman diliminde (öteleme olmadan) depolar. Bu nedenle; 'datetime', zamanı kullanan fonksiyonları yürütmek için UTC zamanında oluşturulmalıdır. MetaTrader 5'ten elde edilen veriler UTC zamanına sahiptir, ancak Python bu verileri yazdırmaya çalışırken yerel zaman ötelemesini tekrar uygular. Bu nedenle, elde edilen veriler görsel sunum için de düzeltilmelidir.

Örnek:

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
```

```
# elde edilen verileri tablo şeklinde görüntülemek için 'pandas' modülünü içe aktar
import pandas as pd
pd.set_option('display.max_columns', 500) # görüntülenecek sütun sayısı
pd.set_option('display.width', 1500)      # görüntülenecek maksimum tablo genişliği
# zaman dilimi ile çalışmak için pytz modülünü içe aktar
import pytz

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# zaman dilimini UTC olarak ayarla
timezone = pytz.timezone("Etc/UTC")
# yerel zaman dilimi ötelemesinin gerçekleşmesini önlemek için UTC zaman diliminde 'da
utc_from = datetime(2020, 1, 10, tzinfo=timezone)
utc_to = datetime(2020, 1, 11, tzinfo=timezone)
# 11.01.2020 - 11.01.2020 aralığındaki AUDUSD tiklerini talep et
ticks = mt5.copy_ticks_range("AUDUSD", utc_from, utc_to, mt5.COPY_TICKS_ALL)
print("Elde edilen tik sayısı:",len(ticks))

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()

# her bir tikle ilgili verileri yeni bir satırda göster
print("Elde edilen tikleri 'olduğu gibi' görüntüle")
count = 0
for tick in ticks:
    count+=1
    print(tick)
    if count >= 10:
        break

# elde edilen verilerden DataFrame oluştur
ticks_frame = pd.DataFrame(ticks)
# saniye cinsinden zamanı datetime biçimine dönüştür
ticks_frame['time']=pd.to_datetime(ticks_frame['time'], unit='s')

# display data
print("\nTikleri içeren veri çerçevesini görüntüle")
print(ticks_frame.head(10))
```

Sonuç:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Elde edilen tik sayısı: 37008
Elde edilen tikleri 'olduğu gibi' görüntüle
(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.)

Tikleri içeren veri çerçevesini görüntüle
                 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
```

Ayrıca bakınız

CopyRates, copy_rates_from_pos, copy_rates_range, copy_ticks_from, copy_ticks_range

orders_total

Aktif emirlerin sayısını elde eder.

```
orders_total()
```

Geri dönüş değeri

Tam sayı değeri.

Not

Fonksiyon, OrdersTotal'a benzerdir.

Örnek:

```
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# aktif emirlerin varlığını kontrol et
orders=mt5.orders_total()
if orders>0:
    print("Total orders=",orders)
else:
    print("Orders not found")

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()
```

Ayrıca bakınız

orders_get, positions_total

orders_get

Sembol veya etikete göre filtreleme özelliğiyle aktif emirleri elde eder. Üç çağrı seçeneği vardır.

Parametresiz çağrı. Tüm sembollerdeki aktif emirler geri döndürülür.

```
orders_get()
```

Aktif emirlerin elde edileceği sembolü belirten çağrı.

```
orders_get(
   symbol="SYMBOL"      // sembol adı
)
```

Aktif emirlerin elde edileceği sembol grubunu belirten çağrı.

```
orders_get(
   group="GROUP"        // emirleri sembollere göre seçmek için filtre
)
```

Emir etiketini belirten çağrı.

```
orders_get(
   ticket=TICKET        // etiket
)
```

symbol="SYMBOL"

[in] Sembol adı. Opsiyonel adlandırılmış parametre. Bir sembol belirtilirse, *ticket* parametresi yoksayılır.

group="GROUP"

[in] Gerekli sembollerden oluşan bir grup düzenlemek için filtre. Opsiyonel adlandırılmış parametre. Bir grup belirtilirse, fonksiyon yalnızca sembol adı için belirtilen ölçütleri karşılayan aktif emirleri geri döndürür.

ticket=TICKET

[in] Emir etiketi (ORDER_TICKET). Opsiyonel adlandırılmış parametre.

Geri dönüş değeri

Bilgiler adlandırılmış bir veri grubu yapısı (namedtuple) biçiminde geri döndürülür. Bir hata durumunda None geri döndürülür. Hata ile ilgili bilgiler last_error() kullanılarak elde edilebilir.

Not

Fonksiyon, OrdersTotal ve OrderSelect ikilisine benzer şekilde tek bir çağrıda tüm aktif emirlerin elde edilmesine olanak sağlar

group parametresi, emirleri sembollere göre filtrelemenize olanak tanır. '*' bir dizgenin başında ve sonunda kullanılabilir.

group parametresi, virgülle ayrılmış birkaç koşul içerebilir. Bir koşul '*' kullanılarak maske olarak ayarlanabilir. Hariç tutma için '!' mantıksal olumsuzlama sembolü kullanılabilir. Tüm koşullar sırayla uygulanır, yani önce gruba dahil etme koşullarını ve ardından dışlama koşulunu belirtmelisiniz. Örneğin, *group*="*, !EUR", önce tüm semboller için emirlerin seçilmesi ve daha sonra sembol adlarında "EUR" içerenlerin hariç tutulması gerektiği anlamına gelir.

Örnek:

```
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # görüntülenecek sütun sayısı
pd.set_option('display.width', 1500)      # görüntülenecek maksimum tablo genişliği
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# GBPUSD'deki aktif emirlerle ilgili verileri görüntüle
orders=mt5.orders_get(symbol="GBPUSD")
if orders is None:
    print("No orders on GBPUSD, error code={}".format(mt5.last_error()))
else:
    print("Total orders on GBPUSD:",len(orders))
    # tüm aktif emirleri görüntüle
    for order in orders:
        print(order)
print()

# adlarında "*GBP*" içeren sembollerdeki emirleri elde et
gbp_orders=mt5.orders_get(group="*GBP*")
if gbp_orders is None:
    print("No orders with group=\"*GBP*\", error code={}".format(mt5.last_error()))
else:
    print("orders_get(group=\"*GBP*\")={}".format(len(gbp_orders)))
    # bu emirleri pandas.DataFrame kullanarak bir tablo halinde göster
    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)

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()

Sonuç:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Total orders on 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
```

Ayrıca bakınız

orders_total, positions_get

order_calc_margin

Belirtilen alım-satım işlemini gerçekleştirmek için hesap para birimi cinsinden marjini geri döndürür.

```
order_calc_margin(
   action,      // emir tipi (ORDER_TYPE_BUY veya ORDER_TYPE_SELL)
   symbol,      // sembol adı
   volume,      // hacim
   price        // açılış fiyatı
   )
```

Parametreler

action

[in] ORDER_TYPE sayımından değer alan emir tipi. Gerekli adsız parametre.

symbol

[in] Finansal enstrüman adı. Gerekli adsız parametre.

volume

[in] İşlem hacmi. Gerekli adsız parametre.

price

[in] Açılış fiyatı. Gerekli adsız parametre.

Geri dönüş değeri

Başarılı olursa gerçek değer, aksi takdirde None geri döner. Hataya ilişkin bilgiler last_error() kullanılarak elde edilebilir.

Not

Fonksiyon, mevcut bekleyen emirler ve açık pozisyonlar dikkate alınmaksızın mevcut hesapta ve mevcut piyasa ortamında belirli bir emir tipi için gerekli marjın miktarının hesaplanmasına olanak sağlar. Fonksiyon, OrderCalcMargin'e benzerdir.

ORDER_TYPE

ID	Açıklama
ORDER_TYPE_BUY	Piyasa alış emri
ORDER_TYPE_SELL	Piyasa satış emri
ORDER_TYPE_BUY_LIMIT	Bekleyen Buy Limit emri
ORDER_TYPE_SELL_LIMIT	Bekleyen Sell Limit emri
ORDER_TYPE_BUY_STOP	Bekleyen Buy Stop emri
ORDER_TYPE_SELL_STOP	Bekleyen Sell Stop emri
ORDER_TYPE_BUY_STOP_LIMIT	Emir fiyatına ulaşıldığında, bekleyen Buy Limit emri StopLimit fiyatından yerleştirilir

ID	Açıklama
ORDER_TYPE_SELL_STOP_LIMIT	Emir fiyatına ulaşıldığında, bekleyen Sell Limit emri StopLimit fiyatından yerleştirilir
ORDER_TYPE_CLOSE_BY	Bir pozisyonu zıt yönde bir pozisyonla kapatma emri

Örnek:

```
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# hesap para birimini elde et
account_currency=mt5.account_info().currency
print("Account currency:",account_currency)

# sembol listesini oluştur
symbols=("EURUSD","GBPUSD","USDJPY", "USDCHF","EURJPY","GBPJPY")
print("Symbols to check margin:", 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,"not found, skipped")
        continue
    if not symbol_info.visible:
        print(symbol, "is not visible, trying to switch on")
        if not mt5.symbol_select(symbol,True):
            print("symbol_select({}}) failed, skipped",symbol)
            continue
    ask=mt5.symbol_info_tick(symbol).ask
    margin=mt5.order_calc_margin(action,symbol,lot,ask)
    if margin != None:
        print("   {} buy {} lot margin: {} {}".format(symbol,lot,margin,account_currer
    else:
        print("order_calc_margin failed: , error code =", mt5.last_error())

# MetaTrader 5 terminaline olan bağlantıyı kapat
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
```

Ayrıca bakınız

order_calc_profit, order_check

order_calc_profit

Belirtilen alım-satım işlemi için hesap para birimi cinsinden kârı geri döndürür.

```
order_calc_profit(
   action,          // emir tipi (ORDER_TYPE_BUY veya ORDER_TYPE_SELL)
   symbol,          // sembol adı
   volume,          // hacim
   price_open,      // açılış fiyatı
   price_close      // kapanış fiyatı
   );
```

Parametreler

action

[in] Emir tipi iki ORDER_TYPE sayım değerinden biri olabilir: ORDER_TYPE_BUY veya ORDER_TYPE_SELL. Gerekli adsız parametre.

symbol

[in] Finansal enstrüman adı. Gerekli adsız parametre.

volume

[in] İşlem hacmi. Gerekli adsız parametre.

price_open

[in] Açılış fiyatı. Gerekli adsız parametre.

price_close

[in] Kapanış fiyatı. Gerekli adsız parametre.

Geri dönüş değeri

Başarılı olursa gerçek değer, aksi takdirde None geri döner. Hataya ilişkin bilgiler last_error() kullanılarak elde edilebilir.

Not

Fonksiyon, mevcut hesapta ve mevcut alım-satım ortamında bir alım-satım işleminin sonucunun hesaplanmasına olanak sağlar. Fonksiyon, OrderCalcProfit'e benzerdir.

Örnek:

```
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
```

```
    quit()

# hesap para birimini elde et
account_currency=mt5.account_info().currency
print("Account currency:",account_currency)

# sembol listesini oluştur
symbols = ("EURUSD","GBPUSD","USDJPY")
print("Symbols to check margin:", symbols)
# alış ve satış için karı hesapla
lot=1.0
distance=300
for symbol in symbols:
    symbol_info=mt5.symbol_info(symbol)
    if symbol_info is None:
        print(symbol,"not found, skipped")
        continue
    if not symbol_info.visible:
        print(symbol, "is not visible, trying to switch on")
        if not mt5.symbol_select(symbol,True):
            print("symbol_select({}}) failed, skipped",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: profit on {} points => {} {}".format(symbol,lot,dista
    else:
        print("order_calc_profit(ORDER_TYPE_BUY) failed, error code =",mt5.last_error
    sell_profit=mt5.order_calc_profit(mt5.ORDER_TYPE_SELL,symbol,lot,bid,bid-distance*
    if sell_profit!=None:
        print("   sell {} {} lots: profit on {} points => {} {}".format(symbol,lot,dis
    else:
        print("order_calc_profit(ORDER_TYPE_SELL) failed, error code =",mt5.last_erro
    print()

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()
```

Sonuç:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

Account currency: USD
Symbols to check margin: ('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
```

Ayrıca bakınız

order_calc_margin, order_check

order_check

Gerekli alım-satım işleminin gerçekleştirilmesi için fon yeterliliğini kontrol eder. Kontrol sonucu MqlTradeCheckResult yapısı olarak geri döndürülür.

```
order_check(
   request      // istek yapısı
   );
```

Parametreler

request

[in] Gerekli alım-satım eylemini tanımlayan MqlTradeRequest tipi yapısı. Gerekli adsız parametre. Talebin doldurulması ve sayım içeriği için bir örnek aşağıda verilmiştir.

Geri dönüş değeri

Kontrol sonucu MqlTradeCheckResult yapısı olarak geri döndürülür. Yanıttaki *request* alanı, order_check()'e iletilen bir alım-satım talebinin yapısını içerir.

Not

Bir talebin başarılı bir şekilde gönderilmesi, **talep edilen alım-satım işleminin başarıyla yürütüleceğinin kanıtı değildir**. order_check fonksiyonu, OrderCheck'e benzerdir.

TRADE_REQUEST_ACTIONS

ID	Açıklama
TRADE_ACTION_DEAL	Belirtilen parametrelerle anında işlem için bir emir yerleştirir (bir piyasa emri yerleştirir)
TRADE_ACTION_PENDING	Belirtilen koşullarda bir işlem gerçekleştirmek için bir emir yerleştirir (bekleyen emir)
TRADE_ACTION_SLTP	Açık pozisyonun Kar Al ve Zararı Durdur seviyelerini değiştirir
TRADE_ACTION_MODIFY	Daha önce yerleştirilmiş olan alım-satım emrinin parametrelerini değiştirir
TRADE_ACTION_REMOVE	Daha önce yerleştirilmiş olan bekleyen emri kaldırır
TRADE_ACTION_CLOSE_BY	Bir pozisyonu zıt yönde bir pozisyonla kapatır

ORDER_TYPE_FILLING

ID	Açıklama
ORDER_FILLING_FOK	Bu gerçekleştirme politikası, bir emrin yalnızca belirtilen hacimde gerçekleştirilebileceği anlamına gelir. Finansal enstrüman için piyasada gerekli miktarda hacim mevcut değilse, emir yürütülmez. İstenen hacim mevcut birkaç teklif fiyatını içerebilir.

ID	Açıklama
ORDER_FILLING_IOC	Bu politika emirde belirtilen hacim dahilinde piyasadaki kullanılabilir maksimum hacimde bir işlemin yapılmasına izin vermek anlamına gelir. Emir tam olarak yerine getirilemezse, kullanılabilir hacime sahip bir emir yürütülür ve kalan hacim iptal edilir.
ORDER_FILLING_RETURN	Bu politika yalnızca piyasa (ORDER_TYPE_BUY ve ORDER_TYPE_SELL), limit ve stop limit emirleri (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT ve ORDER_TYPE_SELL_STOP_LIMIT) ve sadece Piyasa Fiyatından veya Exchange işlem gerçekleştirme modlarına sahip semboller için kullanılır. Kısmen yerine getirilirse, kalan hacme sahip piyasa veya limit emri iptal edilmez ve geçerliliğini korur. ORDER_TYPE_BUY_STOP_LIMIT ve ORDER_TYPE_SELL_STOP_LIMIT emirlerinin aktifleştirilmesi sırasında, uygun bir ORDER_TYPE_BUY_LIMIT/ORDER_TYPE_SELL_LIMIT limit emri ORDER_FILLING_RETURN tipiyle oluşturulur.

ORDER_TYPE_TIME

ID	Açıklama
ORDER_TIME_GTC	Emir, elle iptal edilene kadar kuyrukta kalacak
ORDER_TIME_DAY	Emir sadece mevcut işlem günü içerisinde geçerli kalacak
ORDER_TIME_SPECIFIED	Emir belirtilen tarihe kadar geçerli kalacak
ORDER_TIME_SPECIFIED_DAY	Emir, belirtilen günün 23:59:59'una kadar geçerli kalacak. Eğer bu zaman işlem oturumunun dışındaysa, emir sona erme tarihine en yakın işlem saatinde gerçekleştirilir.

Örnek:

```
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# hesap para birimini elde et
account_currency=mt5.account_info().currency
print("Account currency:",account_currency)

# istek yapısını hazırla
symbol="USDJPY"
symbol_info = mt5.symbol_info(symbol)
```

```
if symbol_info is None:
    print(symbol, "not found, can not call order_check()")
    mt5.shutdown()
    quit()

# sembol MarketWatch'te yoksa, ekle
if not symbol_info.visible:
    print(symbol, "is not visible, trying to switch on")
    if not mt5.symbol_select(symbol,True):
        print("symbol_select({}}) failed, exit",symbol)
        mt5.shutdown()
        quit()

# talebi hazırla
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,
}

# kontrolü yap ve sonucu 'olduğu gibi' görüntüle
result = mt5.order_check(request)
print(result);
# sonucu sözlük olarak talep et ve tek tek görüntüle
result_dict=result._asdict()
for field in result_dict.keys():
    print("   {}={}".format(field,result_dict[field]))
    # eğer bu bir alım-satım talebi yapısı ise, onu da tek tek görüntüle
    if field=="request":
        traderequest_dict=result_dict[field]._asdict()
        for tradereq_filed in traderequest_dict:
            print("       traderequest: {}={}".format(tradereq_filed,traderequest_dict

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()
```

```
Sonuç:
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
```

Ayrıca bakınız

order_send,OrderCheck, Alım-satım işlem tipleri, Alım-satım isteği yapısı, Sonuç kontrolü isteğinin yapısı, Alım-satım isteği sonucunun yapısı

order_send

Bir alım-satım işlemi gerçekleştirmek için terminalden işlem sunucusuna bir istek gönderir. Fonksiyon, OrderSend'e benzerdir.

```
order_send(
   request      // istek yapısı
   );
```

Parametreler

request

[in] Gerekli alım-satım eylemini tanımlayan MqlTradeRequest tipi yapısı. Gerekli adsız parametre. Talebin doldurulması ve sayım içeriği için bir örnek aşağıda verilmiştir.

Geri dönüş değeri

İşlem gerçekleştirme sonucu MqlTradeResult yapısı olarak geri döndürülür. Yanıttaki *request* alanı, order_send()'e iletilen bir alım-satım talebinin yapısını içerir.

MqlTradeRequest işlem istek yapısı

Alan	Açıklama
action	İşlem tipi. Değer, TRADE_REQUEST_ACTIONS sayımının değerlerinden biri olabilir.
magic	Uzman Danışman ID'si. Alım-satım emirlerinin analitik olarak işlenmesine olanak tanır. Her bir uzman danışman bir işlem isteği gönderirken benzersiz bir ID belirleyebilir.
order	Emir etiketi. Bekleyen emirleri değiştirmek için gereklidir.
symbol	Emrin yerleştirileceği alım-satım enstrümanının adı. Emirleri değiştirmek ve pozisyonları kapatmak için gerekli değildir.
volume	İstenen işlem hacmi (lot cinsinden). Bir işlem yaparken gerçek hacim, emir gerçekleştirme tipine bağlıdır.
price	Bir emrin gerçekleştirilmesi gereken fiyat. TRADE_ACTION_DEAL tipine sahip "Piyasa Fiyatından İşlem Gerçekleştirme" (SYMBOL_TRADE_EXECUTION_MARKET) tipindeki enstrümanlara ilişkin piyasa emirleri için fiyat belirlenmez.
stoplimit	Fiyat 'price' değerine ulaştığında, bekleyen Limit emrinin yerleştirildiği fiyat (bu koşul zorunludur). Bekleyen emir o ana kadar alım-satım sistemine geçirilmez.
sl	Fiyat olumsuz yönde hareket ettiğinde Zararı Durdur emrinin etkinleştirileceği fiyat
tp	Fiyat olumlu yönde hareket ettiğinde Kar Al emrinin etkinleştirileceği fiyat
deviation	İstenen fiyattan kabul edilebilir maksimum sapma (puan cinsinden belirtilir)
type	Emir tipi. Değer, ORDER_TYPE sayımının değerlerinden biri olabilir.
type_filling	Emrin gerçekleştirilme tipi. Değer, ORDER_TYPE_FILLING değerlerinden biri olabilir.

Alan	Açıklama
type_time	Emrin sona erme tipi. Değer, ORDER_TYPE_TIME değerlerinden biri olabilir.
expiration	Bekleyen emrin sona erme zamanı (TIME_SPECIFIED tipi emirler için)
comment	Emre yorum ekleme
position	Pozisyon etiketi. Açık bir şekilde tanımlanması için pozisyonu değiştirirken ve kapatırken doldurun. Genellikle, pozisyonu açan emrin etiketi ile aynıdır.
position_by	Zıt pozisyonun etiketi. Bir pozisyonu zıt yönde bir pozisyonla kapatırken kullanılır (aynı sembolde fakat zıt yönde açılır).

Not

Bir alım-satım isteği, işlem sunucusunda birkaç doğrulama aşamasından geçer. İlk olarak, gerekli tüm *istek* alanlarının geçerliliği kontrol edilir. Hata yoksa, sunucu ileri işleme için emri kabul eder. Alım-satım işlemlerinin gerçekleştirilmesiyle ilgili ayrıntılar için OrderSend fonksiyonunun açıklamasına bakın.

Örnek:

```
import time
import MetaTrader5 as mt5

# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ", mt5.__author__)
print("MetaTrader5 package version: ", mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# alış istek yapısını hazırla
symbol = "USDJPY"
symbol_info = mt5.symbol_info(symbol)
if symbol_info is None:
    print(symbol, "not found, can not call order_check()")
    mt5.shutdown()
    quit()

# sembol MarketWatch'te yoksa, ekle
if not symbol_info.visible:
    print(symbol, "is not visible, trying to switch on")
    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,
}

# işlem isteği gönder
result = mt5.order_send(request)
# işlem gerçekleştirme sonucunu kontrol et
print("1. order_send(): by {} {} lots at {} with deviation={} points".format(symbol,lo
if result.retcode != mt5.TRADE_RETCODE_DONE:
    print("2. order_send failed, retcode={}".format(result.retcode))
   # sonucu sözlük olarak talep et ve tek tek görüntüle
    result_dict=result._asdict()
    for field in result_dict.keys():
        print("   {}={}".format(field,result_dict[field]))
        # eğer bu bir alım-satım talebi yapısı ise, onu da tek tek görüntüle
        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 done, ", result)
print("   opened position with POSITION_TICKET={}".format(result.order))
print("   sleep 2 seconds before closing position #{}".format(result.order))
time.sleep(2)
# kapat isteği oluştur
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,
}
# işlem isteği gönder
result=mt5.order_send(request)
# işlem gerçekleştirme sonucunu kontrol et
print("3. close position #{}: sell {} {} lots at {} with deviation={} points".format(p
if result.retcode != mt5.TRADE_RETCODE_DONE:
    print("4. order_send failed, retcode={}".format(result.retcode))
    print("   result",result)
else:
    print("4. position #{} closed, {}".format(position_id,result))
   # sonucu sözlük olarak talep et ve tek tek görüntüle
    result_dict=result._asdict()
    for field in result_dict.keys():
        print("   {}={}".format(field,result_dict[field]))
        # eğer bu bir alım-satım talebi yapısı ise, onu da tek tek görüntüle
        if field=="request":
            traderequest_dict=result_dict[field]._asdict()
            for tradereq_filed in traderequest_dict:
                print("       traderequest: {}={}".format(tradereq_filed,traderequest_

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()

Sonuç:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29
1. order_send(): by USDJPY 0.1 lots at 108.023 with deviation=20 points
2. order_send done,  OrderSendResult(retcode=10009, deal=535084512, order=557416535, v
   opened position with POSITION_TICKET=557416535
   sleep 2 seconds before closing position #557416535
3. close position #557416535: sell USDJPY 0.1 lots at 108.018 with deviation=20 points
4. position #557416535 closed, OrderSendResult(retcode=10009, deal=535084631, order=55
   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
```

Ayrıca bakınız

order_check, OrderSend,Alım-satım işlem tipleri, Alım-satım isteği yapısı, Sonuç kontrolü isteğinin yapısı, Alım-satım isteği sonucunun yapısı

positions_total

Açık pozisyonların sayısını elde eder.

```
positions_total()
```

Geri dönüş değeri

Tam sayı değeri.

Not

Fonksiyon, PositionsTotal'a benzerdir.

Örnek:

```
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# açık pozisyonların varlığını kontrol et
positions_total=mt5.positions_total()
if positions_total>0:
    print("Total positions=",positions_total)
else:
    print("Positions not found")

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()
```

Ayrıca bakınız

positions_get, orders_total

positions_get

Sembol veya etikete göre filtreleme özelliğiyle açık pozisyonları elde eder. Üç çağrı seçeneği vardır.

Parametresiz çağrı. Tüm sembollerdeki açık pozisyonlar geri döndürülür.

```
positions_get()
```

Açık pozisyonların elde edileceği sembolü belirten çağrı.

```
positions_get(
   symbol="SYMBOL"      // sembol adı
)
```

Açık pozisyonların elde edileceği sembol grubunu belirten çağrı.

```
positions_get(
   group="GROUP"        // pozisyonları sembollere göre seçmek için filtre
)
```

Pozisyon etiketini belirten çağrı.

```
positions_get(
   ticket=TICKET        // etiket
)
```

Parametreler

symbol="SYMBOL"

[in] Sembol adı. Opsiyonel adlandırılmış parametre. Bir sembol belirtilirse, *ticket* parametresi yoksayılır.

group="GROUP"

[in] Gerekli sembollerden oluşan bir grup düzenlemek için filtre. Opsiyonel adlandırılmış parametre. Bir grup belirtilirse, fonksiyon yalnızca sembol adı için belirtilen ölçütleri karşılayan pozisyonları geri döndürür.

ticket=TICKET

[in] Pozisyon etiketi (POSITION_TICKET). Opsiyonel adlandırılmış parametre.

Geri dönüş değeri

Bilgiler adlandırılmış bir veri grubu yapısı (namedtuple) biçiminde geri döndürülür. Bir hata durumunda None geri döndürülür. Hata ile ilgili bilgiler last_error() kullanılarak elde edilebilir.

Not

Fonksiyon, PositionsTotal ve PositionSelect ikilisine benzer şekilde tek bir çağrıda tüm açık pozisyonların elde edilmesine olanak sağlar

group parametresi, virgülle ayrılmış birkaç koşul içerebilir. Bir koşul '*' kullanılarak maske olarak ayarlanabilir. Hariç tutma için '!' mantıksal olumsuzlama sembolü kullanılabilir. Tüm koşullar sırayla uygulanır, yani önce gruba dahil etme koşullarını ve ardından dışlama koşulunu belirtmelisiniz. Örneğin, *group*="*, !EUR", önce tüm semboller için pozisyonların seçilmesi ve daha sonra sembol adlarında "EUR" içerenlerin hariç tutulması gerektiği anlamına gelir.

Örnek:

```
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # görüntülenecek sütun sayısı
pd.set_option('display.width', 1500)      # görüntülenecek maksimum tablo genişliği
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# USDCHF'deki açık pozisyonları elde et
positions=mt5.positions_get(symbol="USDCHF")
if positions==None:
    print("No positions on USDCHF, error code={}".format(mt5.last_error()))
elif len(positions)>0:
    print("Total positions on USDCHF =",len(positions))
    # tüm açık pozisyonları görüntüle
    for position in positions:
        print(position)

# adlarında "*USD*" içeren sembollerdeki pozisyonları elde et
usd_positions=mt5.positions_get(group="*USD*")
if usd_positions==None:
    print("No positions with group=\"*USD*\", error code={}".format(mt5.last_error()))
elif len(usd_positions)>0:
    print("positions_get(group=\"*USD*\")={}".format(len(usd_positions)))
    # bu pozisyonları pandas.DataFrame kullanarak bir tablo halinde göster
    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, inp
    print(df)

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()
```

Sonuç:

```
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  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

Ayrıca bakınız

positions_total, orders_get

history_orders_total

İşlem geçmişinden belirtilen aralıktaki emir sayısını elde eder.

```
history_orders_total(
   date_from,     // emirlerin talep edileceği aralığın başlangıç tarihi
   date_to        // emirlerin talep edileceği aralığın bitiş tarihi
   )
```

Parametreler

date_from

[in] İşlemlerin talep edileceği aralığın başlangıç tarihi. 'datetime' nesnesi tarafından veya 1970.01.01'den beri geçen saniye sayısı olarak ayarlanır. Gerekli adsız parametre.

date_to

[in] İşlemlerin talep edileceği aralığın bitiş tarihi. 'datetime' nesnesi tarafından veya 1970.01.01'den beri geçen saniye sayısı olarak ayarlanır. Gerekli adsız parametre.

Geri dönüş değeri

Tam sayı değeri.

Not

Fonksiyon, HistoryOrdersTotal'a benzerdir.

Örnek:

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# geçmişteki emir sayısını elde et
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("Total history orders=",history_orders)
else:
    print("Orders not found in history")

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()
```

Ayrıca bakınız

history_orders_get, history_deals_total

history_orders_get

Pozisyon veya etikete göre filtreleme özelliğiyle işlem geçmişinden emirleri elde eder. Üç çağrı seçeneği vardır.

Zaman aralığını belirten çağrı. Belirtilen aralıktaki tüm emirleri geri döndürür.

```
history_orders_get(
   date_from,                 // emirlerin talep edileceği aralığın başlangıç tarihi
   date_to,                   // emirlerin talep edileceği aralığın bitiş tarihi
   group="GROUP"          // emirleri sembollere göre seçmek için filtre
   )
```

Emir etiketini belirten çağrı. Belirtilen etikete sahip emri geri döndürür.

```
history_orders_get(
   ticket=TICKET          // emir etiketi
)
```

Pozisyon etiketini belirten çağrı. ORDER_POSITION_ID özelliğinde belirtilen pozisyon etiketine sahip tüm emirleri geri döndürür.

```
history_orders_get(
   position=POSITION      // pozisyon etiketi
)
```

Parametreler

date_from

[in] İşlemlerin talep edileceği aralığın başlangıç tarihi. 'datetime' nesnesi tarafından veya 1970.01.01'den beri geçen saniye sayısı olarak ayarlanır. İlk olarak belirtilen gerekli adsız parametre.

date_to

[in] İşlemlerin talep edileceği aralığın bitiş tarihi. 'datetime' nesnesi tarafından veya 1970.01.01'den beri geçen saniye sayısı olarak ayarlanır. İkinci olarak belirtilen gerekli adsız parametre.

group="GROUP"

[in] Gerekli sembollerden oluşan bir grup düzenlemek için filtre. Opsiyonel adlandırılmış parametre. Bir grup belirtilirse, fonksiyon yalnızca sembol adı için belirtilen ölçütleri karşılayan emirleri geri döndürür.

ticket=TICKET

[in] Elde edilecek emir etiketi. Opsiyonel parametre. Belirtilmezse, filtre uygulanmaz.

position=POSITION

[in] Pozisyon etiketi (ORDER_POSITION_ID'de saklanır). Bu pozisyon etiketine sahip tüm emirler elde edilir. Opsiyonel parametre. Belirtilmezse, filtre uygulanmaz.

Geri dönüş değeri

Bilgiler adlandırılmış bir veri grubu yapısı (namedtuple) biçiminde geri döndürülür. Bir hata durumunda None geri döndürülür. Hata ile ilgili bilgiler last_error() kullanılarak elde edilebilir.

Not

Fonksiyon, HistoryOrdersTotal ve HistoryOrderSelect ikilisine benzer şekilde tek bir çağrıda belirli bir aralıktaki tüm geçmiş emirlerin elde edilmesine olanak sağlar.

group parametresi, virgülle ayrılmış birkaç koşul içerebilir. Bir koşul '*' kullanılarak maske olarak ayarlanabilir. Hariç tutma için '!' mantıksal olumsuzlama sembolü kullanılabilir. Tüm koşullar sırayla uygulanır, yani önce gruba dahil etme koşullarını ve ardından dışlama koşulunu belirtmelisiniz. Örneğin, *group*="*, !EUR", önce tüm semboller için işlemlerin seçilmesi ve daha sonra sembol adlarında "EUR" içerenlerin hariç tutulması gerektiği anlamına gelir.

Örnek:

```
from datetime import datetime
import MetaTrader5 as mt5
import pandas as pd
pd.set_option('display.max_columns', 500) # görüntülenecek sütun sayısı
pd.set_option('display.width', 1500)      # görüntülenecek maksimum tablo genişliği
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# geçmişteki emir sayısını elde et
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("No history orders with group=\"*GBP*\", error code={}".format(mt5.last_erro
elif len(history_orders)>0:
    print("history_orders_get({}, {}, group=\"*GBP*\")={}".format(from_date,to_date,le
print()

# tüm geçmiş emirleri pozisyon etiketine göre göster
position_id=530218319
position_history_orders=mt5.history_orders_get(position=position_id)
if position_history_orders==None:
    print("No orders with position #{}".format(position_id))
    print("error code =",mt5.last_error())
elif len(position_history_orders)>0:
    print("Total history orders on position #{}: {}".format(position_id,len(position_h
   # belirtilen pozisyon etiketine sahip tüm geçmiş emirleri göster
    for position_order in position_history_orders:
        print(position_order)
    print()
    # bu emirleri pandas.DataFrame kullanarak bir tablo halinde göster
```

```
    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)

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()

Sonuç:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  5.0.29

history_orders_get(2020-01-01 00:00:00, 2020-03-25 17:17:32.058795, group="*GBP*")=14

Total history orders on position #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
```

Ayrıca bakınız

history_deals_total, history_deals_get

history_deals_total

İşlem geçmişinden belirtilen aralıktaki işlem sayısını elde eder.

```
history_deals_total(
   date_from,     // işlemlerin talep edileceği aralığın başlangıç tarihi
   date_to        // işlemlerin talep edileceği aralığın bitiş tarihi
   )
```

Parametreler

date_from

[in] İşlemlerin talep edileceği aralığın başlangıç tarihi. 'datetime' nesnesi tarafından veya 1970.01.01'den beri geçen saniye sayısı olarak ayarlanır. Gerekli adsız parametre.

date_to

[in] İşlemlerin talep edileceği aralığın bitiş tarihi. 'datetime' nesnesi tarafından veya 1970.01.01'den beri geçen saniye sayısı olarak ayarlanır. Gerekli adsız parametre.

Geri dönüş değeri

Tam sayı değeri.

Not

Fonksiyon, HistoryDealsTotal'a benzerdir.

Örnek:

```
from datetime import datetime
import MetaTrader5 as mt5
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)

# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# geçmişteki işlem sayısını elde et
from_date=datetime(2020,1,1)
to_date=datetime.now()
deals=mt5.history_deals_total(from_date, to_date)
if deals>0:
    print("Total deals=",deals)
else:
    print("Deals not found in history")

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()
```

Ayrıca bakınız

history_deals_get, history_orders_total

history_deals_get

Pozisyon veya etikete göre filtreleme özelliğiyle işlem geçmişinden belirtilen aralıktaki işlemleri elde eder.

Zaman aralığını belirten çağrı. Belirtilen aralıktaki tüm işlemleri geri döndürür.

```
history_deals_get(
   date_from,                // işlemlerin talep edileceği aralığın başlangıç tarihi
   date_to,                  // işlemlerin talep edileceği aralığın bitiş tarihi
   group="GROUP"          // işlemleri sembollere göre seçmek için filtre
   )
```

Emir etiketini belirten çağrı. DEAL_ORDER özelliğinde belirtilen emir etiketine sahip tüm işlemleri geri döndürür.

```
history_deals_get(
   ticket=TICKET         // emir etiketi
)
```

Pozisyon etiketini belirten çağrı. DEAL_POSITION_ID özelliğinde belirtilen pozisyon etiketine sahip tüm işlemleri geri döndürür.

```
history_deals_get(
   position=POSITION     // pozisyon etiketi
)
```

Parametreler

date_from

[in] İşlemlerin talep edileceği aralığın başlangıç tarihi. 'datetime' nesnesi tarafından veya 1970.01.01'den beri geçen saniye sayısı olarak ayarlanır. İlk olarak belirtilen gerekli adsız parametre.

date_to

[in] İşlemlerin talep edileceği aralığın bitiş tarihi. 'datetime' nesnesi tarafından veya 1970.01.01'den beri geçen saniye sayısı olarak ayarlanır. İkinci olarak belirtilen gerekli adsız parametre.

group="GROUP"

[in] Gerekli sembollerden oluşan bir grup düzenlemek için filtre. Opsiyonel adlandırılmış parametre. Bir grup belirtilirse, fonksiyon yalnızca sembol adı için belirtilen ölçütleri karşılayan işlemleri geri döndürür.

ticket=TICKET

[in] Emir etiketi (DEAL_ORDER'da saklanır). Bu emir etiketine sahip tüm işlemler elde edilir. Opsiyonel parametre. Belirtilmezse, filtre uygulanmaz.

position=POSITION

[in] Pozisyon etiketi (DEAL_POSITION_ID'de saklanır). Bu pozisyon etiketine sahip tüm işlemler elde edilir. Opsiyonel parametre. Belirtilmezse, filtre uygulanmaz.

Geri dönüş değeri

Bilgiler adlandırılmış bir veri grubu yapısı (namedtuple) biçiminde geri döndürülür. Bir hata durumunda None geri döndürülür. Hata ile ilgili bilgiler last_error() kullanılarak elde edilebilir.

Not

Fonksiyon, HistoryDealsTotal ve HistoryDealSelect ikilisine benzer şekilde tek bir çağrıda belirli bir aralıktaki tüm geçmiş işlemlerin elde edilmesine olanak sağlar.

group parametresi, işlemleri sembollere göre filtrelemenize olanak tanır. '*' bir dizgenin başında ve sonunda kullanılabilir.

group parametresi, virgülle ayrılmış birkaç koşul içerebilir. Bir koşul '*' kullanılarak maske olarak ayarlanabilir. Hariç tutma için '!' mantıksal olumsuzlama sembolü kullanılabilir. Tüm koşullar sırayla uygulanır, yani önce gruba dahil etme koşullarını ve ardından dışlama koşulunu belirtmelisiniz. Örneğin, *group*="*, !EUR", önce tüm semboller için işlemlerin seçilmesi ve daha sonra sembol adlarında "EUR" içerenlerin hariç tutulması gerektiği anlamına gelir.

Örnek:

```
import MetaTrader5 as mt5
from datetime import datetime
import pandas as pd
pd.set_option('display.max_columns', 500) # görüntülenecek sütun sayısı
pd.set_option('display.width', 1500)      # görüntülenecek maksimum tablo genişliği
# MetaTrader 5 paketi ile ilgili verileri görüntüle
print("MetaTrader5 package author: ",mt5.__author__)
print("MetaTrader5 package version: ",mt5.__version__)
print()
# MetaTrader 5 terminaline bağlantı kur
if not mt5.initialize():
    print("initialize() failed, error code =",mt5.last_error())
    quit()

# geçmişteki işlem sayısını elde et
from_date=datetime(2020,1,1)
to_date=datetime.now()
# adlarında "GBP" içeren semboller için belirtilen aralıktaki işlemleri elde et
deals=mt5.history_deals_get(from_date, to_date, group="*GBP*")
if deals==None:
    print("No deals with group=\"*USD*\", error code={}".format(mt5.last_error()))
elif len(deals)> 0:
    print("history_deals_get({}, {}, group=\"*GBP*\")={}".format(from_date,to_date,le

# adlarında "EUR" veya "GBP" içermeyen sembollerdeki işlemleri elde et
deals = mt5.history_deals_get(from_date, to_date, group="*,!*EUR*,!*GBP*")
if deals == None:
    print("No deals, error code={}".format(mt5.last_error()))
elif len(deals) > 0:
    print("history_deals_get(from_date, to_date, group=\"*,!*EUR*,!*GBP*\") =", len(d
    # elde edilen işlemleri 'olduğu gibi' görüntüle
    for deal in deals:
```

```
        print("  ",deal)
    print()
    # bu işlemleri pandas.DataFrame kullanarak bir tablo halinde göster
    df=pd.DataFrame(list(deals),columns=deals[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    print(df)
print("")

# pozisyon #530218319 ile ilgili tüm işlemleri elde et
position_id=530218319
position_deals = mt5.history_deals_get(position=position_id)
if position_deals == None:
    print("No deals with position #{}".format(position_id))
    print("error code =", mt5.last_error())
elif len(position_deals) > 0:
    print("Deals with position id #{}: {}".format(position_id, len(position_deals)))
    # bu işlemleri pandas.DataFrame kullanarak bir tablo halinde göster
    df=pd.DataFrame(list(position_deals),columns=position_deals[0]._asdict().keys())
    df['time'] = pd.to_datetime(df['time'], unit='s')
    print(df)

# MetaTrader 5 terminaline olan bağlantıyı kapat
mt5.shutdown()

Sonuç:
MetaTrader5 package author:  MetaQuotes Software Corp.
MetaTrader5 package version:  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 with position id #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

Ayrıca bakınız

history_deals_total, history_orders_get

Makine Öğreniminde ONNX Modelleri

Açık sinir ağı santrali (Open Neural Network Exchange, ONNX), makine öğrenimi modelleri için açık kaynaklı bir formattır. Bu projenin birkaç önemli avantajı vardır:

- ONNX, Microsoft, Facebook ve Amazon gibi büyük şirketler tarafından desteklenmektedir.
- Açık format olması, farklı makine öğrenimi araç takımları arasında model dönüştürmeyi mümkün kılarken, Microsoft'un ONNXMLTools aracı da modellerin ONNX formatına dönüştürülmesine olanak sağlar.
- MQL5, iletilen parametre türü modelle eşleşmediğinde, model girdileri ve çıktıları için otomatik veri türü dönüştürme sağlar.
- ONNX modelleri, çeşitli makine öğrenimi araçları kullanılarak oluşturulabilir. Şu anda Caffe2, Microsoft Cognitive Toolkit, MXNet, PyTorch ve OpenCV'de desteklenmektedirler. Diğer popüler çerçeveler ve kütüphaneler için de arayüzler mevcuttur.
- MQL5 dili, ONNX modelini ticaret stratejinize uygulamanıza ve böylece onu finansal piyasalarda etkin bir şekilde ticaret işlemleri gerçekleştirmeyi sağlayan MetaTrader 5 platformunun tüm avantajlarıyla birlikte kullanmanıza olanak tanır.
- Modeli gerçek ticarette kullanmadan önce, model davranışını üçüncü taraf araçlar olmadan doğrudan strateji sınayıcıda geçmiş veriler üzerinde test edebilirsiniz.

MQL5, ONNX ile çalışmak için aşağıdaki fonksiyonları sunmaktadır:

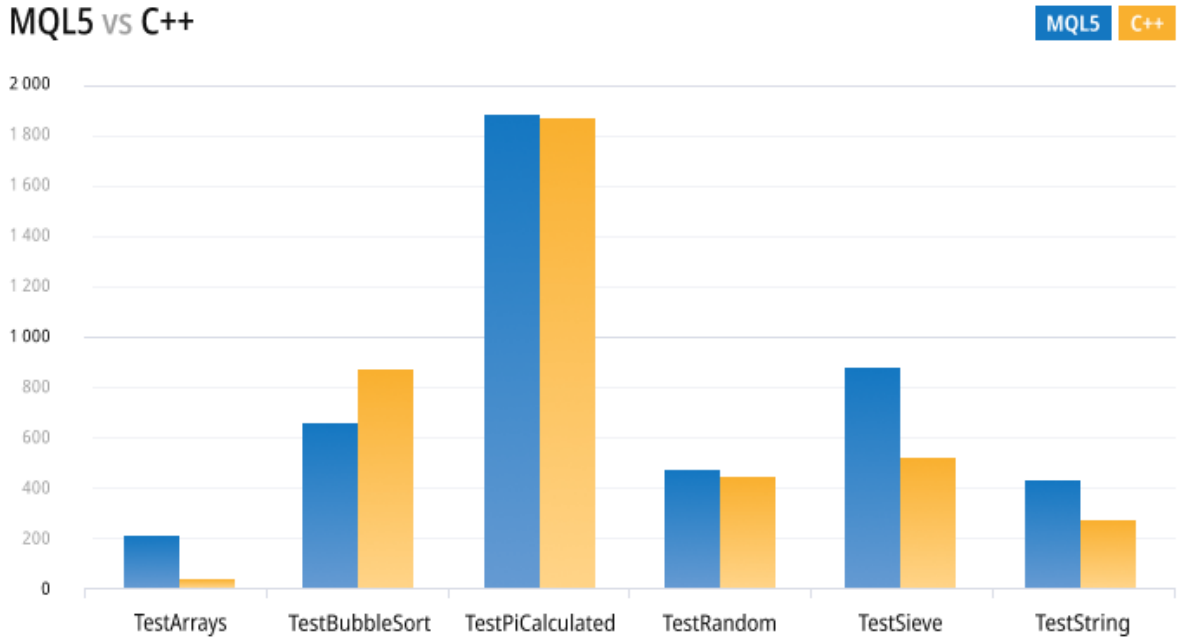
Fonksiyon	Eylem
OnnxCreate	*.onnx dosyasından model yükleyerek ONNX oturumu oluşturur
OnnxCreateFromBuffer	Veri dizisinden model yükleyerek ONNX oturumu oluşturur
OnnxRelease	ONNX oturumunu kapatır
OnnxRun	ONNX modelini çalıştırır
OnnxGetInputCount	ONNX modelindeki girdi sayısını alır
OnnxGetOutputCount	ONNX modelindeki çıktı sayısını alır
OnnxGetInputName	İndekse göre modelin girdisinin adını alır
OnnxGetOutputName	İndekse göre modelin çıktısının adını alır
OnnxGetInputTypeInfo	Modelden girdi türünün tanımını alır
OnnxGetOutputTypeInfo	Modelden çıktı türünün tanımını alır
OnnxSetInputShape	Modelin girdi verilerinin şeklini indekse göre ayarlar
OnnxSetOutputShape	Modelin çıktı verilerinin şeklini indekse göre ayarlar

MQL5'te ONNX Desteği

ONNX, makine öğrenimi modellerini temsil etmek için tasarlanmış açık kaynaklı bir formattır. Bu standart, geliştiricilerin modelleri farklı çerçeveler, araçlar, çalışma zamanları ve derleyicilerle kullanmasına olanak sağlayan ortak bir operatör kümesi ve dosya formatı tanımlar.

Böylece, açık kaynaklı ONNX formatı, makine öğrenimi modellerinin farklı platformlar ve makine öğrenimi araç takımları arasında aktarılmasına olanak tanır. Yapay zeka geliştiricilerinin oluşturulan modelleri MetaTrader 5 platformu tarafından sunulan yüksek performanslı yürütme ortamında çalıştırabilmeleri için MQL5 diline ONNX desteği eklenmiştir.

MQL5'in yürütme hızı, C++ uygulamalarının yürütme hızıyla karşılaştırılabilir düzeydedir. Bu, MQL5 ve C++ üzerinde yürütülen standart testlerin sonuçlarıyla kanıtlanmıştır. Çubuk ne kadar düşükse, yürütme için o kadar az zaman (milisaniye cinsinden) harcanmaktadır ve sonuç o kadar iyi olmaktadır. Testler, Windows 10 (yapı 17763) x64, Xeon E5-2630 v4 @ 2,20GHz, Bellek: 65457 Mb üzerinde gerçekleştirilmiştir.



Yeni asenkronize ticaret işlemleri ve yerel ONNX desteği, daha önceden yalnızca az sayıda profesyonel yapay zeka geliştiricisi ve kurumsal yatırımcı için mevcut olan yeni fırsatları hizmetinize sunmaktadır. MQL5'teki ONNX desteği, yatırımcıların tercih ettikleri geliştirme ortamında finansal piyasa ticareti için modeller eğitmesine ve ardından düşük ağ maliyetleri, yüksek emir defteri güncelleme hızları ve asenkronize emir sunumuyla ticaret yapmasına olanak tanır.

Şu anda ONNX, bu açık kaynak projenin daha da geliştirilmesini garanti eden Microsoft, Facebook ve Amazon gibi ortak şirketler tarafından geliştirilmekte ve sürdürülmektedir.

Model Dönüştürme

ONNX, farklı makine öğrenimi araç takımlarından modellerin kullanılmasına olanak tanıyan açık kaynaklı bir formattır. Bu format, Chainer, Caffee2 ve PyTorch gibi birçok çerçeve tarafından desteklenmektedir.

Modelleri ONNX formatına dönüştürmek için en popüler araçlardan biri Microsoft'un ONNXMLTools aracıdır.

ONNXMLTools'u kurma ve kullanma talimatları GitHub deposu nda bulunmaktadır. Şu anda aşağıdaki araç takımları desteklenmektedir:

- Keras (keras2onnx dönüştürücü sarmalayıcısı)
- Tensorflow (tf2onnx dönüştürücü sarmalayıcısı)
- scikit-learn (skl2onnx dönüştürücü sarmalayıcısı)
- Apple Core ML
- Spark ML (deneysel)
- LightGBM
- libscm;
- XGBoost;
- H2O
- CatBoost

ONNXMLTools kolayca kurulabilir. Kurulum ayrıntıları ve model dönüştürme örnekleri için lütfen https://github.com/onnx/onnxmltools#install adresindeki proje sayfasına bakın.

ONNX Modellerini Çalıştırırken Girdi ve Çıktı Değerlerini Otomatik Dönüştürme

MQL5'teki mevcut ONNX sürümü, girdi/çıktı değerleri olarak yalnızca tensörleri desteklemektedir. Tensörler, aşağıdaki veri türlerindeki elemanlara sahip veri dizileridir:

ONNX türü	Karşılık gelen MQL5 türü
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	—

ONNX modellerine girdi/çıktı değerleri olarak yalnızca diziler, vektörler veya matrisler (bunlara **Veriler** olarak değineceğiz) beslenebilir.

İletilen parametre türü, ONNX modelinin parametre türüyle eşleşmiyorsa ve OnnxRun, ONNX_NO_CONVERSION bayrağı belirtilmeden çağrıldıysa, otomatik veri dönüştürme uygulanacaktır. Otomatik dönüştürme, ONNX modeli çalıştırılmadan önce, kullanıcı **Veriler**inin ilgili dönüştürmeyle ONNX tensörlerine kopyalanacağı anlamına gelir.

ONNX modeli otomatik dönüştürme olmadan çalıştırıldığında, model herhangi bir ek kopyalama olmaksızın **Veriler** kullanılarak hesaplanacaktır.

> **ÖNEMLİ!** Otomatik dönüştürme aşırılığı (truncate) kontrol etmez, bu nedenle ONNX modeline girdi olarak kullanılan verileri ve türlerini dikkatle izlemelisiniz.

Otomatik dönüştürme, aşağıdaki ONNX türlerini desteklemektedir:

- 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

Desteklenmeyen türler:

- ONNX_DATA_TYPE_BFLOAT16
- ONNX_DATA_TYPE_STRING

Tensör türlerine göre otomatik dönüştürme kuralları

MQL5 türü, model tarafından desteklenen türler listesinde yer almıyorsa, ONNX modelinin çalıştırılması ERR_ONNX_NOT_SUPPORTED hatası (hata kodu 5802) geri döndürecektir.

Not: Otomatik dönüştürme sırasında, color türü uint olarak, datetime türü de long olarak işlenir.

Girdi değerlerini otomatik dönüştürme

ONNX türü (tensör elemanı türü)	Otomatik dönüştürme tarafından desteklenen MQL5 türü
ONNX_DATA_TYPE_BOOL	bool, char, uchar, short, ushort, int, color, uint, datetime, long, float, double, complex Dönüştürme sırasında **Veri** elemanları, 0'a karşı basit bir karşılaştırmayla kontrol edilir
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	ONNX_DATA_TYPE_FLOAT'a bakın
ONNX_DATA_TYPE_INT8	ONNX_DATA_TYPE_FLOAT'a bakın

ONNX türü (tensör elemanı türü)	Otomatik dönüştürme tarafından desteklenen MQL5 türü
ONNX_DATA_TYPE_UINT16	ONNX_DATA_TYPE_FLOAT'a bakın
ONNX_DATA_TYPE_INT16	ONNX_DATA_TYPE_FLOAT'a bakın
ONNX_DATA_TYPE_INT32	ONNX_DATA_TYPE_FLOAT'a bakın
ONNX_DATA_TYPE_INT64	ONNX_DATA_TYPE_FLOAT'a bakın
ONNX_DATA_TYPE_DOUBLE	ONNX_DATA_TYPE_FLOAT'a bakın
ONNX_DATA_TYPE_UINT32	ONNX_DATA_TYPE_FLOAT'a bakın
ONNX_DATA_TYPE_UINT64	ONNX_DATA_TYPE_FLOAT'a bakın
ONNX_DATA_TYPE_COMPLEX64	complex
ONNX_DATA_TYPE_COMPLEX128	complex

Çıktı değerlerini otomatik dönüştürme

ONNX türü (tensör elemanı türü)	Otomatik dönüştürme tarafından desteklenen MQL5 türü
ONNX_DATA_TYPE_BOOL	bool, char, uchar, short, ushort, int, color, uint, datetime, long, float, double, complex Tensör elemanı sıfırsa, Veri elemanı 0 olarak ayarlanır; aksi takdirde, değer 1'dir
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	ONNX_DATA_TYPE_FLOAT'a bakın
ONNX_DATA_TYPE_INT8	ONNX_DATA_TYPE_FLOAT'a bakın
ONNX_DATA_TYPE_UINT16	ONNX_DATA_TYPE_FLOAT'a bakın
ONNX_DATA_TYPE_INT16	ONNX_DATA_TYPE_FLOAT'a bakın
ONNX_DATA_TYPE_INT32	ONNX_DATA_TYPE_FLOAT'a bakın
ONNX_DATA_TYPE_INT64	ONNX_DATA_TYPE_FLOAT'a bakın
ONNX_DATA_TYPE_DOUBLE	ONNX_DATA_TYPE_FLOAT'a bakın
ONNX_DATA_TYPE_UINT32	ONNX_DATA_TYPE_FLOAT'a bakın
ONNX_DATA_TYPE_UINT64	ONNX_DATA_TYPE_FLOAT'a bakın
ONNX_DATA_TYPE_COMPLEX64	complex
ONNX_DATA_TYPE_COMPLEX128	complex

Ayrıca bakınız

Tür Dönüştürme

Model Oluşturma

ONNX formatında hazır bir model elde etmek için birçok yöntem mevcuttur. Popüler ONNX Model Zoo kütüphanesi, farklı görev türleri için önceden eğitilmiş birkaç ONNX modeli içerir. Bu koleksiyonun avantajı, her modelin not defterinin eğitim veri kümesine linkler ve model mimarisini açıklayan orijinal belgeye referanslar içermesidir.

Çoğu makine öğrenimi çerçevesi Python'ı kullanır. Python için ONNX çalışma zamanını kurmak adına aşağıdaki komutlardan birini kullanın:

```
pip install onnxruntime        # CPU build
pip install onnxruntime-gpu    # GPU build
```

Python'da ONNX çalışma zamanını çağırmak için aşağıdaki komutu kullanın:

```
import onnxruntime
session = onnxruntime.InferenceSession("path to model")
```

Model girdileri ve çıktıları için ilgili modelin dokümantasyonuna bakın. Modeli görüntülemek adına Netron veya WinML Dashboard gibi görselleştirme araçlarını kullanabilirsiniz. ONNX çalışma zamanında, modelin meta verilerini, girdilerini ve çıktılarını sorgulayabilirsiniz:

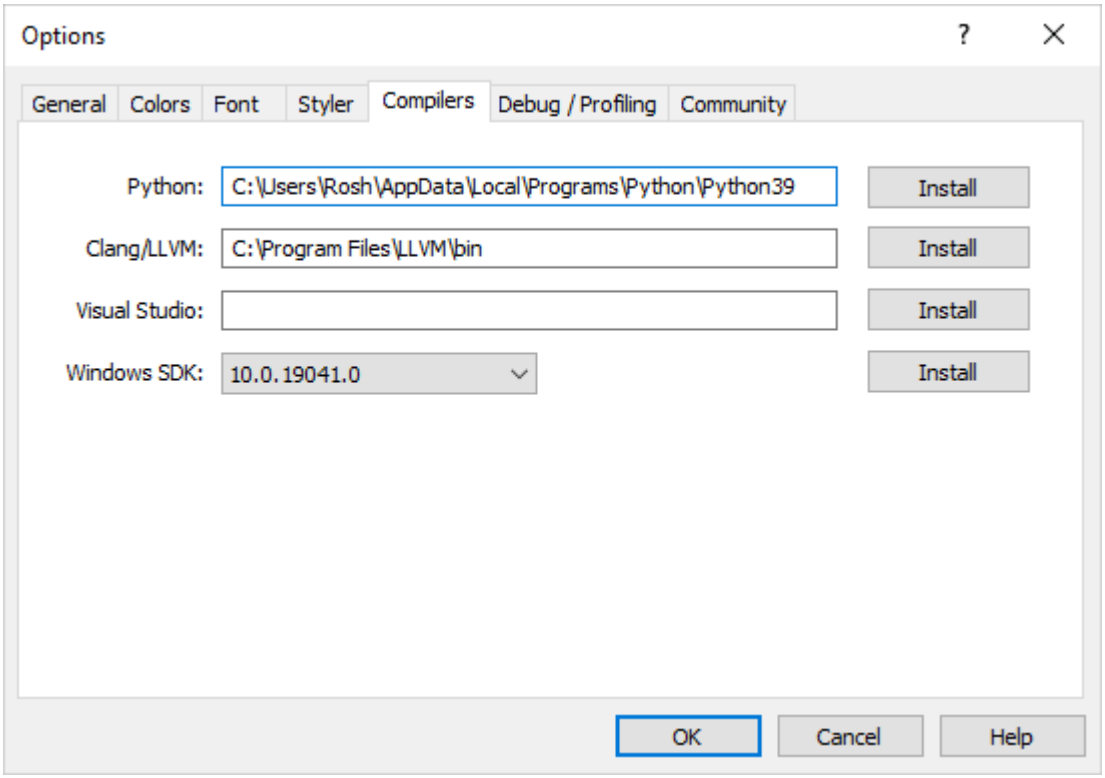
```
results = session.run(["output1", "output2"], {
                      "input1": indata1, "input2": indata2})
results = session.run([], {"input1": indata1, "input2": indata2})
```

ONNX modellerini Python'ı kullanarak doğrudan MetaTrader 5 terminalinde veya MetaEditor'da oluşturabilirsiniz.

MetaTrader 5'te Python

MetaTrader 5, Python komut dosyaları için kullanıma hazır destek sunmaktadır. Bu amaçla, terminal geliştiricileri Python için MetaTrader5 modülü sağlamıştır: https://pypi.org/project/MetaTrader5.

MetaEditor entegre geliştirme ortamı, yalnızca MQL5'te uygulama yazmanıza değil, aynı zamanda Python komut dosyalarını doğrudan düzenleyiciden çalıştırmanıza da olanak tanır. Bunu yapmak için MetaEditor ayarlarından yürütülebilir dosyanın yolunu belirtin:



Python bilgisayarınızda kurulu değilse, kurulum dosyasını indirmek için Kura tıklayın.

MetaEditor'da bir Python komut dosyası oluşturabilir veya onu terminalin veri klasörüne aktarabilir ve F7 (Derle) tuşunu kullanarak hemen çalıştırabilirsiniz. Devamında MetaTrader 5 terminali açılacak ve komut dosyası mevcut grafikte başlatılacaktır. Python konsolundan (stdout, stderr) gelen mesajlar Hatalar bölümü altında görüntülenecektir.

MetaTrader 5'te modellerle çalışma

MQL5 dili, ONNX modellerini doğrudan MetaTrader 5 terminalinde çalıştırmanıza olanak sağlar. Bu üç adımda yapılır:

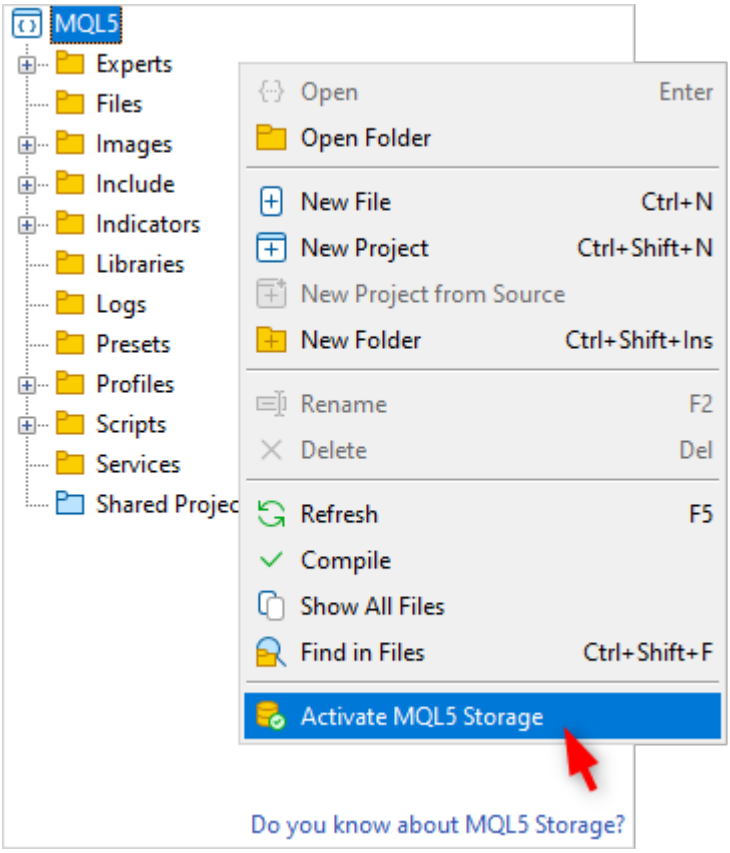
1. Modeli Python gibi üçüncü taraf bir platformda eğitin
2. Modeli ONNX'e dönüştürün
3. ONNX modelini, ONNX fonksiyonunu kullanarak bir Uzman Danışmana dahil edin ve MetaTrader 5 terminalinde çalıştırın

MQL5 dilinin Python entegrasyonu, Python komut dosyasının çalıştırılmasına ve ONNX modelinin MetaEditor'da kaydedilmesine veya doğrudan MetaTrader 5'te grafik üzerinde çalıştırılmasına olanak tanır. Modeli, önceden yazılmış bir Python komut dosyası kullanarak doğrudan terminalde ihtiyaç duyduğunuz sıklıkta eğitebilirsiniz. Kütüphane, fiyat verilerini elde etmek için ONNX modeline girdi olarak kullanılabilecek hazır fonksiyonlar içerir:

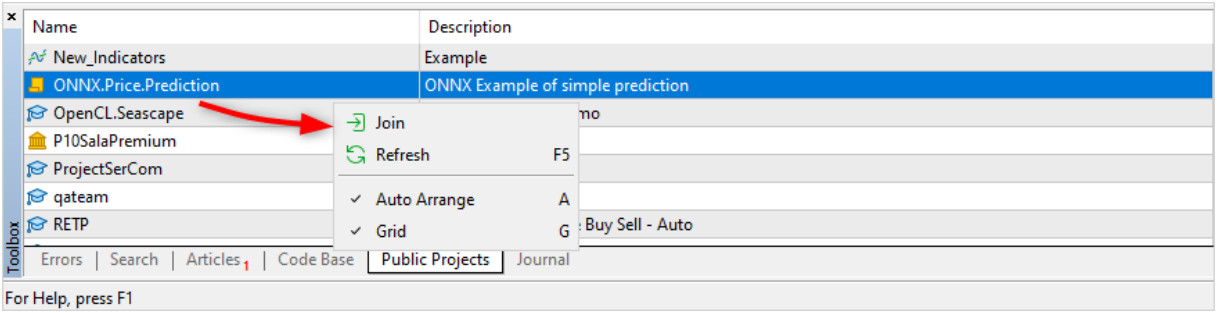
- copy_rates_from - belirtilen tarihten itibaren çubukları alır
- copy_rates_from_pos - belirtilen indeksten itibaren çubukları alır
- copy_rates_range - belirtilen tarih aralığı için çubukları alır
- copy_ticks_from - belirtilen tarihten itibaren tikleri alır
- copy_ticks_range - belirtilen tarih aralığı için tikleri alır

Model örneği

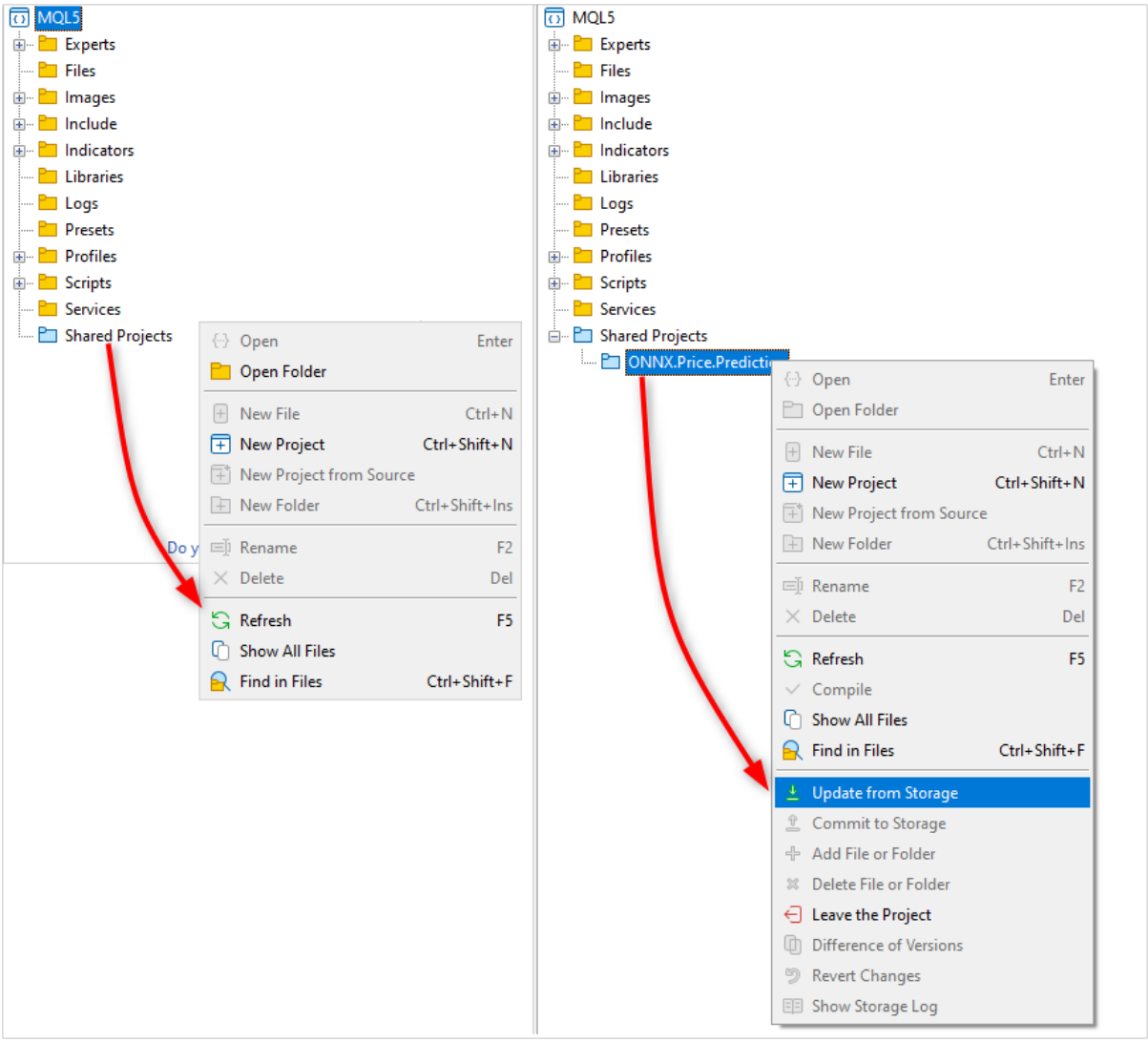
Herkese açık projelerde hazır bir ONNX modeli örneği bulunmaktadır. İlk olarak, MetaEditor ayarlarından MQL5 giriş kimliğinizi (büyük/küçük harfe duyarlı) belirterek Kılavuzdan MQL5 Depoyu etkinleştirmelisiniz.



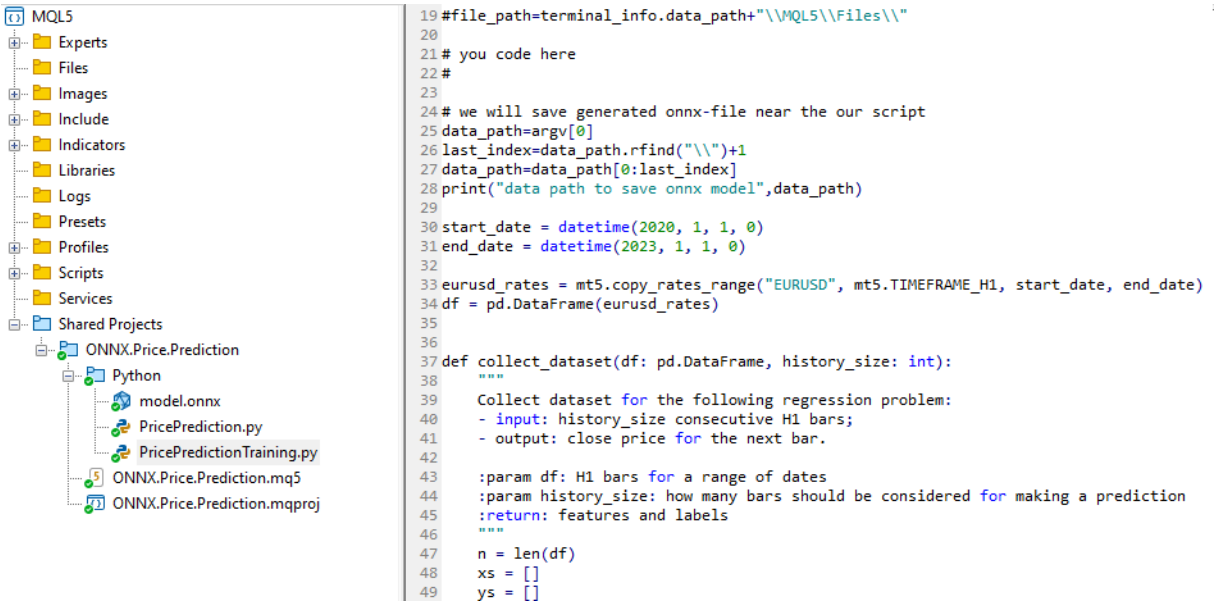
Etkinleştirdikten sonra, ONNX.Price.Prediction projesini bulun ve içerik menüsü komutuyla projeye katılın.



Ardından, projeyi MQL5 Depodan güncelleyin.



Proje bir ONNX modeli, iki Python komut dosyası, projenin çalışması için bir MQL5 komut dosyası ve bir MQL5 proje dosyası (ONNX.Price.Prediction.mqproj) içermektedir.

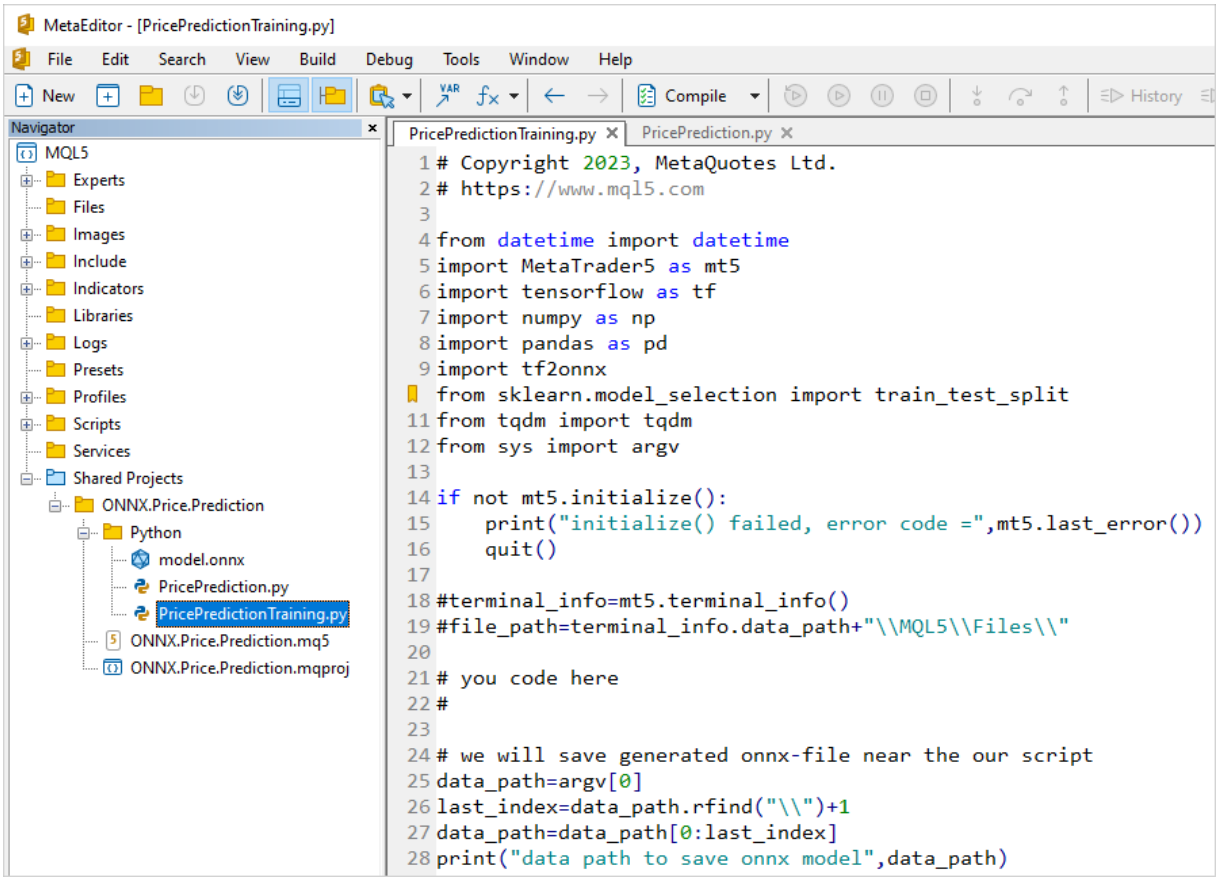



```
#file_path=terminal_info.data_path+"\\MQL5\\Files\\"

# 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 = []
```

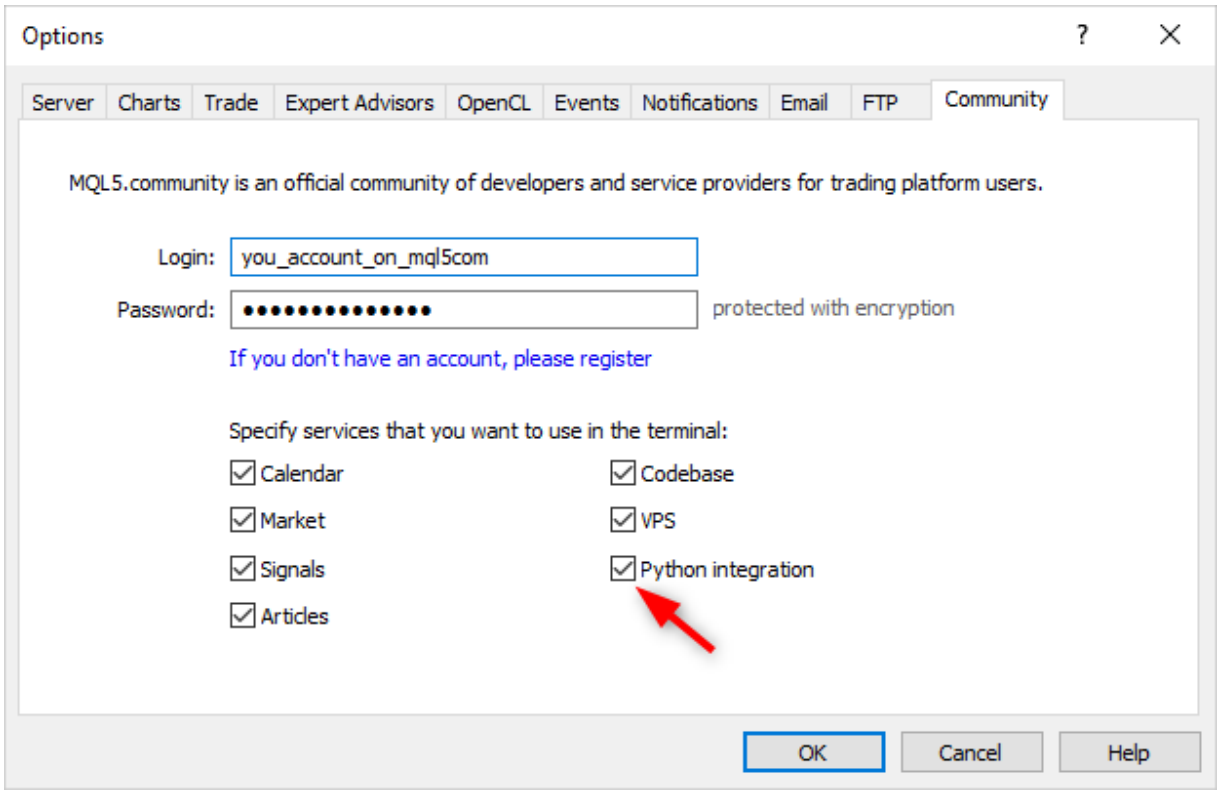
Projede yer alan PricePredictionTraining.py komut dosyasını kullanarak kendiniz bir ONNX modeli oluşturabilirsiniz. Bunun için öncelikle komut satırından gerekli modülleri kurmalısınız.

```
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
```

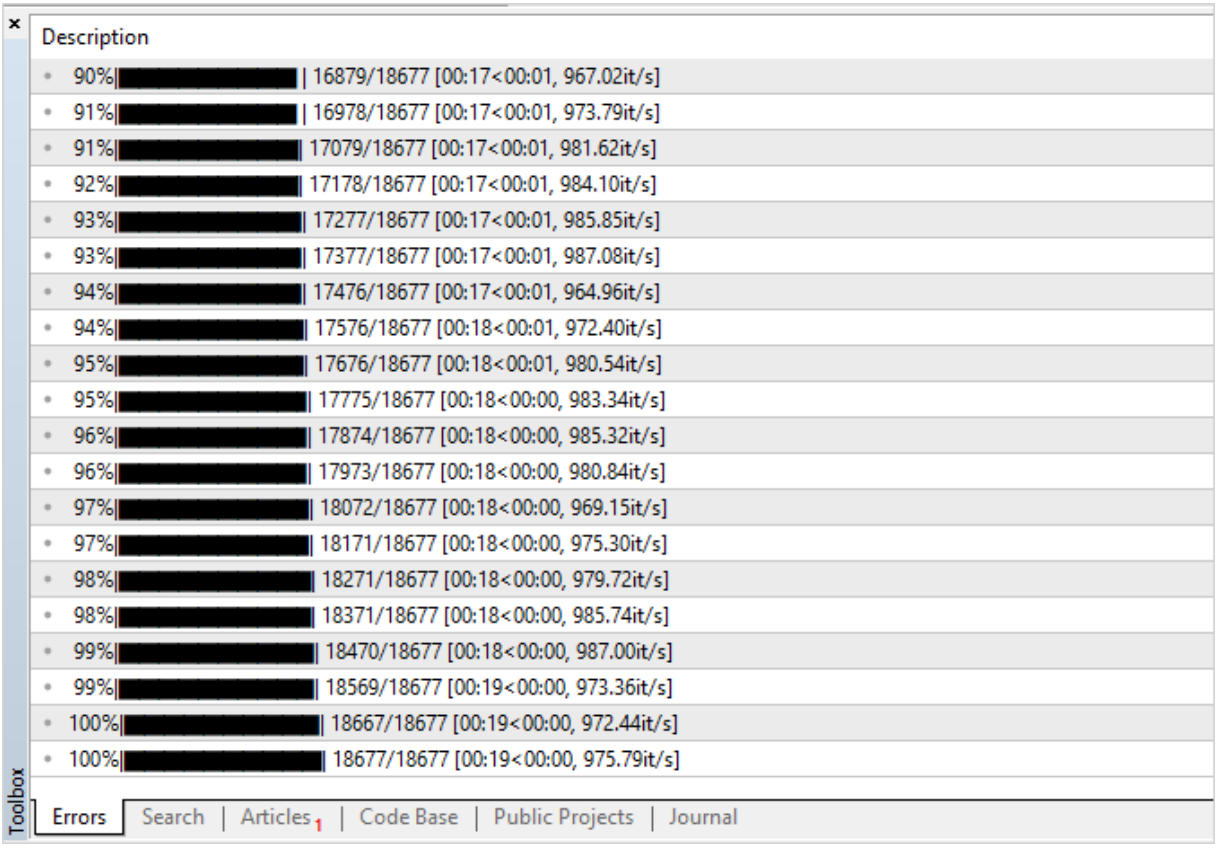
Modülleri kurduktan sonra MetaEditor'da PricePredictionTraining.py komut dosyasını açın ve Derle butonu yada F7 tuşu ile çalıştırın.



Python komut dosyasını çalıştırmadan önce, MetaTrader 5 terminalinin EURUSD sembolünün mevcut olduğu bir sunucuya bağlı olduğundan emin olun. Örneğin, MetaQuotes-Demo sunucusuna bağlanın ve terminal ayarlarından "Python ile entegrasyon" seçeneğini işaretleyin.

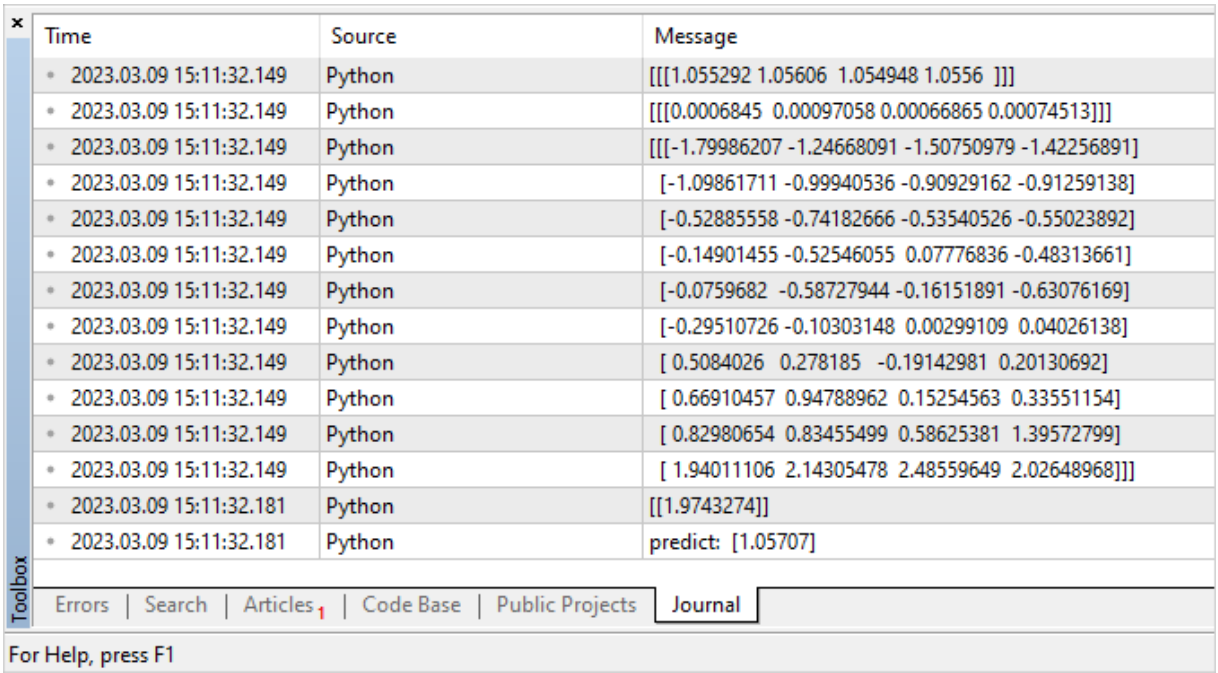


Ağı eğitirken, MetaEditor, eğitim tamamlanana kadar Python komut dosyasından mesajlar yazdıracaktır.



Sonuç %100 olduğunda, ONNX modeli hazırdır ve <terminal veri klasörü>\MQL5\Shared Projects\ONNX.Price.Prediction\Python yolundaki proje klasörüne kaydedilir.

Ortaya çıkan modeli, **F7**'ye basıp ikinci PricePrediction.py komut dosyasını çalıştırarak kontrol edebilirsiniz.



Model Çalıştırma

Bir ONNX modelinin MQL5'te çalıştırılması için şu 3 adımın tamamlanması gerekir:

1. Modeli, OnnxCreate fonksiyonunu kullanarak bir *.onnx dosyasından veya OnnxCreateFromBuffer fonksiyonunu kullanarak bir diziden yükleyin.
2. OnnxSetInputShape ve OnnxSetOutputShape fonksiyonlarını kullanarak girdi ve çıktı verilerinin şekillerini belirtin.
3. Modeli, ilgili girdi ve çıktı parametrelerini ileterek OnnxRun fonksiyonunu kullanarak çalıştırın.
4. Gerektiğinde OnnxRelease fonksiyonunu kullanarak modelin çalışmasını sonlandırabilirsiniz.

Bir ONNX modeli oluştururken, https://github.com/microsoft/onnxruntime/blob/rel-1.14.0/docs/OperatorKernels.md adresinde açıklanan mevcut sınırları ve kısıtlamaları göz önünde bulundurmalısınız.

Bu tür kısıtlamaların bazı örnekleri aşağıda gösterilmektedir:

İşlem	Desteklenen veri türleri
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)

Aşağıda, herkese açık ONNX.Price.Prediction projesinden bir MQL5 kodu örneği bulunmaktadır.

```
const long   ExtOutputShape[] = {1,1};     // modelin çıktı şekli
const long   ExtInputShape [] = {1,10,4}; // modelin girdi şekli
#resource "Python/model.onnx" as uchar ExtModel[]// kaynak olarak model
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   matrix rates;
//--- 10 çubuk al
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,2,10))
      return(-1);
//--- girdi olarak OHLC vektörü kümesi kullan
   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);
//--- normalleştirme matrislerini doldur
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
```

```
     }
//--- girdi verilerini normalleştir
   x_norm-=mm;
   x_norm/=ms;
//--- modeli oluştur
   long handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- girdi verilerinin şeklini belirt
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- çıktı verilerinin şeklini belirt
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- normalleştirilmiş girdi verilerini float türüne dönüştür
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- modelin çıktı verilerini (yani fiyat öngörüsünü) buradan al
   vectorf y_norm(1);
//--- modeli çalıştır
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- modelin çıktı değerini günlüğe yazdır
   Print(y_norm);
//--- öngörülen fiyatı elde etmek için ters dönüşümü yap
   double y_pred=y_norm[0]*s[3]+m[3];
   Print("price predicted:",y_pred);
//--- çalışmayı tamamla
   OnnxRelease(handle);
   return(0);
  }
```

Komut dosyası çalışması örneği:

```
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
```

MetaTrader 5 terminali, hesaplamalar için en uygun yürütücüyü seçti - ONNX Runtime Execution Provider. Bu örnekte, model CPU üzerinde yürütüldü.

Önceki 10 çubuğun değerlerine dayalı olarak yapılan başarılı Kapanış fiyatı öngörülerinin yüzdesini hesaplamak için komut dosyasını düzenleyelim.

```
#resource "Python/model.onnx" as uchar ExtModel[]// kaynak olarak model

#define TESTS 10000  // test veri kümesi sayısı
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
int OnStart()
  {
//--- modeli oluştur
   long session_handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
   if(session_handle==INVALID_HANDLE)
     {
      Print("Cannot create model. Hata ",GetLastError());
      return(-1);
     }

//--- model için girdi tensörü büyüklüğü tanımlanmadığından, açıkça belirtilmelidir
//--- birinci indeks grup büyüklüğüdür, ikinci indeks seri büyüklüğüdür, üçüncü indeks
   const long input_shape[]={1,10,4};
   if(!OnnxSetInputShape(session_handle,0,input_shape))
     {
      Print("OnnxSetInputShape error ",GetLastError());
      return(-2);
     }

//--- model için çıktı tensörü büyüklüğü tanımlanmadığından, açıkça belirtilmelidir
//--- birinci indeks grup büyüklüğüdür, girdi tensöründeki grup büyüklüğüyle eşleşmeli
//--- ikinci indeks öngörülen fiyatların sayısıdır (burada sadece Kapanış fiyatı öngö
   const long output_shape[]={1,1};
   if(!OnnxSetOutputShape(session_handle,0,output_shape))
     {
      Print("OnnxSetOutputShape error ",GetLastError());
      return(-3);
     }
```

```
//--- testleri çalıştır
   vector closes(TESTS);       // kontrol fiyatlarını depolamak için vektör
   vector predicts(TESTS);     // elde edilen öngörüleri depolamak için vektör
   vector prev_closes(TESTS); // önceki fiyatları depolamak için vektör

   matrix rates;               // OHLC serilerini almak için matris
   matrix splitted[2];         // serileri test ve kontrole bölmek için iki alt matris
   ulong  parts[]={10,1};      // bölünmüş alt matrislerin büyüklükleri

//--- önceki çubuktan başla
   for(int i=1; i<=TESTS; i++)
     {
      //--- 11 çubuk al
      rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,i,11);
      //--- matrisi test ve kontrole böl
      rates.Vsplit(parts,splitted);
      //--- kontrol matrisinden Kapanış fiyatını al
      closes[i-1]=splitted[1][3][0];
      //--- test edilen serideki son Kapanış fiyatı
      prev_closes[i-1]=splitted[0][3][9];

      //--- 10 çubukluk test matrisini teste gönder
      predicts[i-1]=PricePredictionTest(session_handle,splitted[0]);
      //--- çalışma zamanı hatası
      if(predicts[i-1]<=0)
        {
         OnnxRelease(session_handle);
         return(-4);
        }
     }
//--- çalışmayı tamamla
   OnnxRelease(session_handle);
//--- fiyat hareketinin doğru öngörülüp öngörülmediğini değerlendir
   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);
  }
//+------------------------------------------------------------------+
//|  Verileri hazırla ve modeli çalıştır                             |
//+------------------------------------------------------------------+
double PricePredictionTest(const long session_handle,matrix& rates)
  {
```

```
   static matrixf input_data(10,4); //dönüştürülmüş girdi için matris
   static vectorf output_data(1);   // sonucu almak için vektör
   static matrix mm(10,4);          // Mean yatay vektörlerin matrisi
   static matrix ms(10,4);          // Std yatay vektörlerin matrisi

//--- modelin girdisi OHLC dikey vektörler kümesi olmalıdır
   matrix x_norm=rates.Transpose();
//--- fiyatları normalleştir
   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;

//--- modeli çalıştır
   input_data.Assign(x_norm);
   if(!OnnxRun(session_handle,ONNX_DEBUG_LOGS,input_data,output_data))
     {
      Print("OnnxRun error ",GetLastError());
      return(0);
     }
//--- fiyatı çıktı değerinden normalleştirilmemiş hale getir
   double y_pred=output_data[0]*s[3]+m[3];

   return(y_pred);
  }
```

Komut dosyasını çalıştıralım: öngörü doğruluğu yaklaşık %51'dir.

```
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 %
```

Strateji Sınayıcıda Model Doğrulama

Finansal piyasalarda ticaret işlemleri gerçekleştirmek için oluşturulan modeller, MetaTrader 5 terminalinde Strateji Sınayıcıda doğrulanabilir. Bu, piyasa ortamını ve ticaret koşullarını ek olarak taklit etme ihtiyacını ortadan kaldıran en hızlı ve en uygun seçenektir.

Modeli test etmek için, herkese açık ONNX.Price.Prediction projesindeki kodu temel alan bir Uzman Danışman oluşturalım. Bu, bazı düzenlemeler gerektirecektir.

Modelin oluşturulmasını OnInit fonksiyonuna, onnx oturumunun kapatılmasını da OnDeinit fonksiyonuna taşıyacağız. Modelle çalışma ana bloğu, OnTick işleyicisine yerleştirilecektir.

Ayrıca, mevcut Kapanış fiyatı ile öngörüyü karşılaştırmak için gerekli olan önceki iki çubuğun Kapanış fiyatlarının elde edilmesini de ekleyelim.

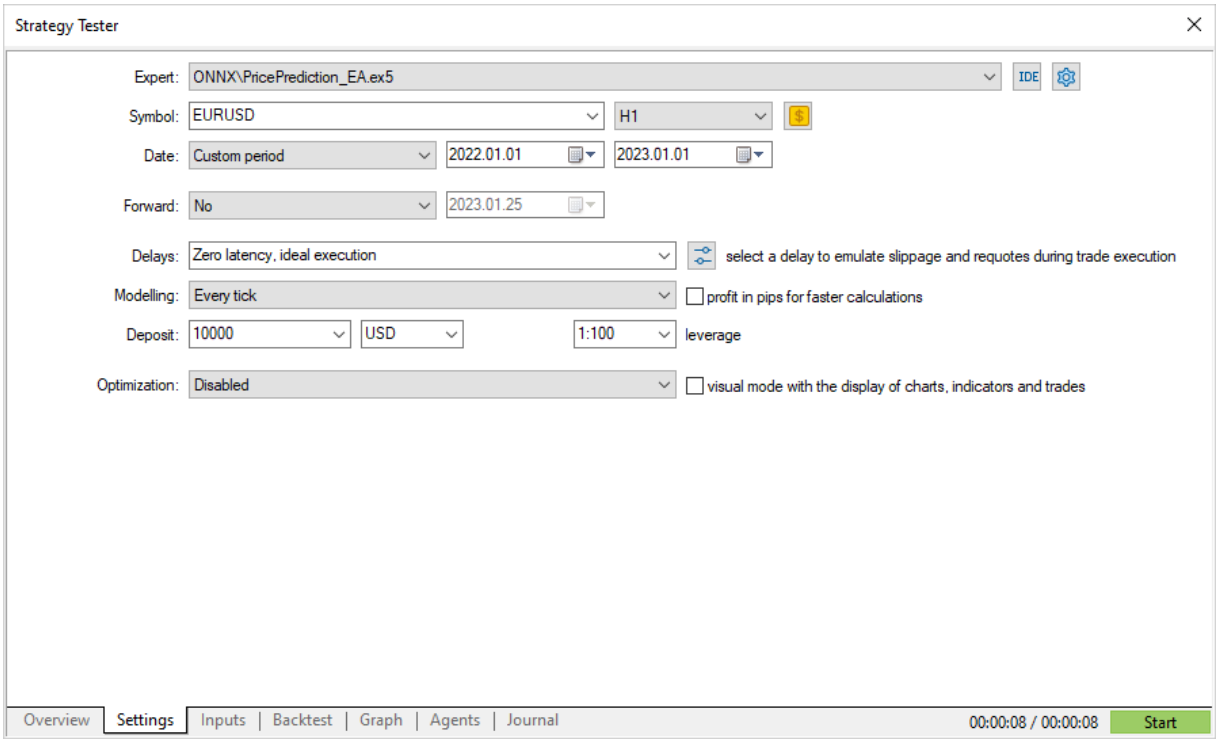
Uzman Danışman kodu kısadır ve okunması kolaydır.

```
const long   ExtInputShape [] = {1,10,4}; // modelin girdi şekli
const long   ExtOutputShape[] = {1,1};    // modelin çıktı şekli
#resource "Python/model.onnx" as uchar ExtModel[];// kaynak olarak model

long handle;          // model tanıtıcısı
ulong predictions=0; // öngörü sayacı
ulong confirmed=0;   // başarılı öngörü sayacı
//+------------------------------------------------------------------+
//| Uzman danışman başlatma fonksiyonu                               |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- temel kontroller
   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);
     }
//--- modeli oluştur
   handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- girdi verilerinin şeklini belirt
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

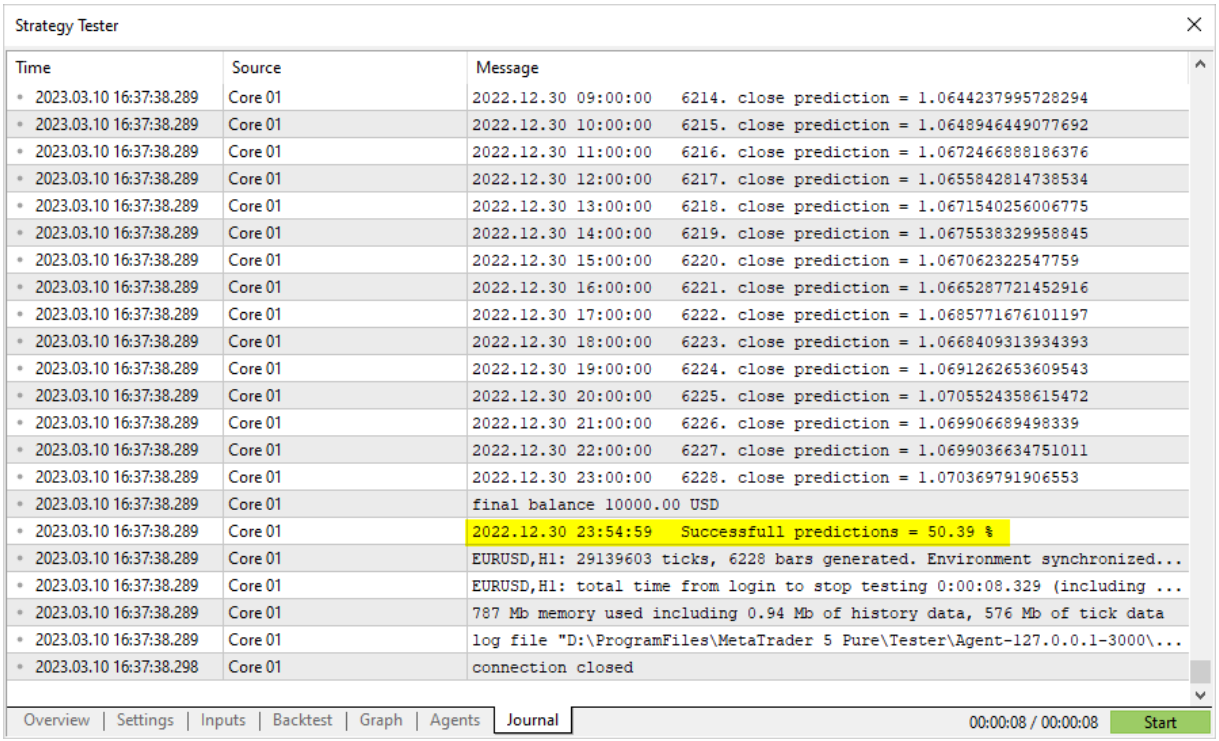
```
//--- çıktı verilerinin şeklini belirt
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//---
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Uzman danışman sonlandırma fonksiyonu                            |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- model çalışmasını tamamla
   OnnxRelease(handle);
//--- öngörü istatistiklerini hesapla ve yazdır
   PrintFormat("Successfull predictions = %.2f %%",confirmed*100./double(predictions));
  }
//+------------------------------------------------------------------+
//| Uzman Danışman tik fonksiyonu                                    |
//+------------------------------------------------------------------+
void OnTick()
  {
   static datetime open_time=0;
   static double predict;
//--- mevcut çubuğun açılış zamanını kontrol et
   datetime time=iTime(_Symbol,_Period,0);
   if(time==0)
     {
      PrintFormat("Failed to get Time(0), error %d", GetLastError());
      return;
     }
//--- açılış zamanı değişmediyse, bir sonraki OnTick çağrısına kadar çıkış yap
   if(time==open_time)
      return;
//--- tamamlanan son iki çubuğun Kapanış fiyatlarını al
   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]; // öngörülen fiyat değişimi
   double delta_actual=close[1]-close[0]; // mevcut fiyat değişimi
   if((delta_predict>0 && delta_actual>0) || (delta_predict<0 && delta_actual<0))
      confirmed++;
```

```
//--- sonraki çubuktaki fiyatı doğrulamak için yeni çubuktaki Kapanış fiyatını hesapla
   matrix rates;
//--- 10 çubuk al
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,1,10))
      return;
//--- girdi olarak OHLC vektörü kümesi kullan
   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);
//--- normalleştirme matrislerini doldur
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//--- girdi verilerini normalleştir
   x_norm-=mm;
   x_norm/=ms;
//--- normalleştirilmiş girdi verilerini float türüne dönüştür
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- modelin çıktı verilerini (yani fiyat öngörüsünü) buradan al
   vectorf y_norm(1);
//--- modeli çalıştır
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
     }
//--- öngörülen fiyatı elde etmek ve yeni çubukta doğrulamak için ters dönüşüm yap
   predict=y_norm[0]*s[3]+m[3];
   predictions++;  // öngörü sayacını artır
   Print(predictions,". close prediction = ",predict);
//--- sonraki tiki kontrol etmek için çubuğun açılış zamanını kaydet
   open_time=time;
  }
```

Uzman Danışmanı derliyoruz ve modelin üzerinde eğitildiği EURUSD sembolünü ve H1 zaman dilimini belirterek 2022 yılı döneminde test ediyoruz. Kod yeni bir çubuğun ortaya çıkışını kontrol ettiğinden tik modelleme modu göz ardı edilebilir.



Başlatıyoruz ve sonucu test günlüğünden görebiliriz. 2022'de öngörülerin %50'den biraz fazlasının doğru olduğunu göstermektedir.



Ön model testi tatmin edici sonuçlar verdiyse, bu modeli temel alan tam teşekküllü bir ticaret stratejisi yazmaya başlayabilirsiniz.

OnnxCreate

*.onnx dosyasından model yükleyerek ONNX oturumu oluşturur.

```
long  OnnxCreate(
   string  filename,  // dosya yolu
   uint    flags      // modeli oluşturmak için bayraklar
   );
```

Parametreler

filename

[in] \MQL5\Files\ klasörüne göre *.onnx model dosyasının yolu.

flags

[in] Model oluşturma modunu tanımlayan ENUM_ONNX_FLAGS numaralandırmasından bayraklar: ONNX_COMMON_FOLDER ve ONNX_DEBUG_LOGS.

Geri dönüş değeri

Oluşturulan oturumun tanıtıcısı veya hata oluşursa INVALID_HANDLE. Hata kodunu almak için GetLastError fonksiyonunu çağırın.

Not

Belirtilen dosya diskte bulunamazsa, sistem ada '.onnx' uzantısını ekleyerek dosyayı açmayı yeniden dener.

OnnxCreateFromBuffer

Veri dizisinden model yükleyerek ONNX oturumu oluşturur.

```
long  OnnxCreateFromBuffer(
   const uchar&  buffer[],    // dizi referansı
   ulong         flags        // model oluşturma bayrakları
   );
```

Parametreler

buffer

[in] ONNX model verilerini içeren dizi.

flags

[in] Model oluşturma modunu tanımlayan ENUM_ONNX_FLAGS numaralandırmasından bayraklar: ONNX_COMMON_FOLDER ve ONNX_DEBUG_LOGS.

Geri dönüş değeri

Oluşturulan oturumun tanıtıcısı veya hata oluşursa INVALID_HANDLE. Hata kodunu almak için GetLastError fonksiyonunu çağırın.

OnnxRelease

ONNX oturumunu kapatır.

```
bool  OnnxRelease(
   long   onnx_handle  // ONNX oturumu tanıtıcısı
   );
```

Parametreler

onnx_handle

[in] OnnxCreate veya OnnxCreateFromBuffer aracılığıyla oluşturulan ONNX oturumu nesnesinin tanıtıcısı.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür. Hata kodunu almak için GetLastError fonksiyonunu çağırın.

OnnxRun

ONNX modelini çalıştırır.

```
bool  OnnxRun(
   long    onnx_handle,  // ONNX oturumu tanıtıcısı
   ulong   flags,        // çalışma modunu tanımlayan bayraklar
   ...                   // modelin girdileri ve çıktıları
   );
```

Parametreler

onnx_handle

[in] OnnxCreate veya OnnxCreateFromBuffer aracılığıyla oluşturulan ONNX oturumu nesnesinin tanıtıcısı.

flags

[in] Çalışma modunu tanımlayan ENUM_ONNX_FLAGS numaralandırmasından bayraklar: ONNX_DEBUG_LOGS ve ONNX_NO_CONVERSION.

...

[in] [out] Model girdileri ve çıktıları.

Başarılı olursa true, aksi takdirde false geri döndürür. Hata kodunu almak için GetLastError fonksiyonunu çağırın.

ENUM_ONNX_FLAGS

Kimlik	Açıklama
ONNX_DEBUG_LOGS	Hata ayıklama günlüklerini yazdırır
ONNX_NO_CONVERSION	Otomatik dönüştürmeyi devre dışı bırakır, kullanıcı verilerini olduğu gibi kullanır
ONNX_COMMON_FOLDER	Common\Files klasöründen model dosyası yükler; değer, FILE_COMMON bayrağına eşittir

Örnek:

```
const long                              ExtOutputShape[] = {1,1};     // modelin çıktı s
const long                              ExtInputShape [] = {1,10,4}; // modelin girdi s
#resource "Python/model.onnx" as uchar ExtModel[]                    // kaynak olarak r
//+------------------------------------------------------------------+
//| Komut dosyası başlatma fonksiyonu                                |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   matrix rates;
//--- 10 çubuk al
```

```
   if(!rates.CopyRates("EURUSD",PERIOD_H1,COPY_RATES_OHLC,2,10))
      return(-1);
//--- girdi olarak OHLC vektörü kümesi kullan
   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);
//--- normalleştirme matrislerini doldur
   for(int i=0; i<10; i++)
     {
      mm.Row(m,i);
      ms.Row(s,i);
     }
//--- girdi verilerini normalleştir
   x_norm-=mm;
   x_norm/=ms;
//--- modeli oluştur
   long handle=OnnxCreateFromBuffer(ExtModel,ONNX_DEBUG_LOGS);
//--- girdi verilerinin şeklini belirt
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("OnnxSetInputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- çıktı verilerinin şeklini belirt
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("OnnxSetOutputShape failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- normalleştirilmiş girdi verilerini float türüne dönüştür
   matrixf x_normf;
   x_normf.Assign(x_norm);
//--- modelin çıktı verilerini (yani fiyat öngörüsünü) buradan al
   vectorf y_norm(1);
//--- modeli çalıştır
   if(!OnnxRun(handle,ONNX_DEBUG_LOGS | ONNX_NO_CONVERSION,x_normf,y_norm))
     {
      Print("OnnxRun failed, error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- modelin çıktı değerini günlüğe yazdır
   Print(y_norm);
//--- öngörülen fiyatı elde etmek için ters dönüşümü yap
   double y_pred=y_norm[0]*s[3]+m[3];
```

```
   Print("price predicted:",y_pred);
//--- çalışmayı tamamla
   OnnxRelease(handle);
   return(0);
  };
```

Ayrıca bakınız

OnnxSetInputShape, OnnxSetOutputShape

OnnxGetInputCount

ONNX modelindeki girdi sayısını alır.

```
long  OnnxGetInputCount(
   long   onnx_handle  // ONNX oturumu tanıtıcısı
   );
```

Parametreler

onnx_handle

[in] OnnxCreate veya OnnxCreateFromBuffer aracılığıyla oluşturulan ONNX oturumu nesnesinin tanıtıcısı.

Geri dönüş değeri

Başarılı olması durumunda girdi parametrelerinin sayısını geri döndürür; aksi takdirde -1 geri döndürür. Hata kodunu almak için GetLastError fonksiyonunu çağırın.

OnnxGetOutputCount

ONNX modelindeki çıktı sayısını alır.

```
long  OnnxGetOutputCount(
   long   onnx_handle  // ONNX oturumu tanıtıcısı
   );
```

Parametreler

onnx_handle

[in] OnnxCreate veya OnnxCreateFromBuffer aracılığıyla oluşturulan ONNX oturumu nesnesinin tanıtıcısı.

Geri dönüş değeri

Başarılı olması durumunda çıktı parametrelerinin sayısını geri döndürür; aksi takdirde -1 geri döndürür. Hata kodunu almak için GetLastError fonksiyonunu çağırın.

OnnxGetInputName

İndekse göre modelin girdisinin adını alır.

```
string  OnnxGetInputName(
   long   onnx_handle,  // ONNX oturumu tanıtıcısı
   long   index         // parametre indeksi
   );
```

Parametreler

onnx_handle

[in] OnnxCreate veya OnnxCreateFromBuffer aracılığıyla oluşturulan ONNX oturumu nesnesinin tanıtıcısı.

index

[in] 0'dan başlayarak girdi parametresinin indeksi.

Geri dönüş değeri

Başarılı olması durumunda girdi parametresinin adını geri döndürür; aksi halde NULL geri döndürür. Hata kodunu almak için GetLastError fonksiyonunu çağırın.

OnnxGetOutputName

İndekse göre modelin çıktısının adını alır.

```
string  OnnxGetOutputName(
   long   onnx_handle,  // ONNX oturumu tanıtıcısı
   long   index         // parametre indeksi
   );
```

Parametreler

onnx_handle

[in] OnnxCreate veya OnnxCreateFromBuffer aracılığıyla oluşturulan ONNX oturumu nesnesinin tanıtıcısı.

index

[in] 0'dan başlayarak çıktı parametresinin indeksi.

Geri dönüş değeri

Başarılı olması durumunda çıktı parametresinin adını geri döndürür; aksi halde NULL geri döndürür. Hata kodunu almak için GetLastError fonksiyonunu çağırın.

OnnxGetInputTypeInfo

Modelden girdi türünün tanımını alır.

```
bool  OnnxGetInputTypeInfo(
   long           onnx_handle,  // ONNX oturumu tanıtıcısı
   long           index,        // parametre indeksi
   OnnxTypeInfo&  typeinfo      // parametre türü tanımı
   );
```

Parametreler

onnx_handle

[in] OnnxCreate veya OnnxCreateFromBuffer aracılığıyla oluşturulan ONNX oturumu nesnesinin tanıtıcısı.

index

[in] 0'dan başlayarak girdi parametresinin indeksi.

typeinfo

[out] Girdi parametresinin türünü tanımlayan OnnxTypeInfo yapısı.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür. Hata kodunu almak için GetLastError fonksiyonunu çağırın.

OnnxGetOutputTypeInfo

Modelden çıktı türünün tanımını alır.

```
bool  OnnxGetOutputTypeInfo(
   long           onnx_handle,  // ONNX oturumu tanıtıcısı
   long           index,        // parametre indeksi
   OnnxTypeInfo&  typeinfo      // parametre türü tanımı
   );
```

Parametreler

onnx_handle

[in] OnnxCreate veya OnnxCreateFromBuffer aracılığıyla oluşturulan ONNX oturumu nesnesinin tanıtıcısı.

index

[in] 0'dan başlayarak çıktı parametresinin indeksi.

typeinfo

[out] Çıktı parametresinin türünü tanımlayan OnnxTypeInfo yapısı.

Geri dönüş değeri

Başarılı olursa true, aksi takdirde false geri döndürür. Hata kodunu almak için GetLastError fonksiyonunu çağırın.

OnnxSetInputShape

Modelin girdi verilerinin şeklini indekse göre ayarlar.

```
bool  OnnxSetInputShape(
   long           onnx_handle,  // ONNX oturumu tanıtıcısı
   long           input_index,  // girdi parametresi indeksi
   const ulong&   shape[]       // girdi verilerinin şeklini tanımlayan dizi
   );
```

Parametreler

onnx_handle

[in] OnnxCreate veya OnnxCreateFromBuffer aracılığıyla oluşturulan ONNX oturumu nesnesinin tanıtıcısı.

input_index

[in] 0'dan başlayarak girdi parametresinin indeksi.

shape

[in] Modelin girdi verilerinin şeklini tanımlayan dizi.

Geri dönüş değeri

Başarılı olması durumunda girdi parametresinin adını geri döndürür; aksi halde NULL geri döndürür. Hata kodunu almak için GetLastError fonksiyonunu çağırın.

Örnek:

```
//---- modelin girdi ve çıktı verilerinin şekillerini tanımla
   const long  ExtOutputShape[] = {1,1};
   const long  ExtInputShape [] = {1,10,4};
//--- modeli oluştur
   long handle=OnnxCreateFromBuffer(model,ONNX_DEBUG_LOGS);
//--- girdi verilerinin şeklini belirt
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("failed, OnnxSetInputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- çıktı verilerinin şeklini belirt
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("failed, OnnxSetOutputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

Ayrıca bakınız

OnnxSetOutputShape

OnnxSetOutputShape

Modelin çıktı verilerinin şeklini indekse göre ayarlar.

```
bool  OnnxSetOutputShape(
   long           onnx_handle,    // ONNX oturumu tanıtıcısı
   long           output_index,   // çıktı parametresi indeksi
   const ulong&   shape[]         // çıktı verilerinin şeklini tanımlayan dizi
   );
```

Parametreler

onnx_handle

[in] OnnxCreate veya OnnxCreateFromBuffer aracılığıyla oluşturulan ONNX oturumu nesnesinin tanıtıcısı.

output_index

[in] 0'dan başlayarak çıktı parametresinin indeksi.

shape

[in] Modelin çıktı verilerinin şeklini tanımlayan dizi.

Geri dönüş değeri

Başarılı olması durumunda girdi parametresinin adını geri döndürür; aksi halde NULL geri döndürür. Hata kodunu almak için GetLastError fonksiyonunu çağırın.

Örnek:

```
//---- modelin girdi ve çıktı verilerinin şekillerini tanımla
   const long  ExtOutputShape[] = {1,1};
   const long  ExtInputShape [] = {1,10,4};
//--- modeli oluştur
   long handle=OnnxCreateFromBuffer(model,ONNX_DEBUG_LOGS);
//--- girdi verilerinin şeklini belirt
   if(!OnnxSetInputShape(handle,0,ExtInputShape))
     {
      Print("failed, OnnxSetInputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
//--- çıktı verilerinin şeklini belirt
   if(!OnnxSetOutputShape(handle,0,ExtOutputShape))
     {
      Print("failed, OnnxSetOutputShape error ",GetLastError());
      OnnxRelease(handle);
      return(-1);
     }
```

Ayrıca bakınız

OnnxSetInputShape

Veri Yapıları

ONNX modelleriyle çalışmak için aşağıdaki veri yapıları kullanılır:

OnnxTypeInfo

Yapı, ONNX modelinin girdi veya çıktı parametresinin türünü tanımlar.

```
struct OnnxTypeInfo
 {
  ENUM_ONNX_TYPE         type;          // parametre türü
  OnnxTensorTypeInfo     tensor;        // tensör tanımı
  OnnxMapTypeInfo        map;           // harita tanımı
  OnnxSequenceTypeInfo   sequence;      // sekans tanımı
 };
```

Girdi olarak yalnızca tensör (ONNX_TYPE_TENSOR) kullanılabilir. Bu durumda, yalnızca OnnxTypeInfo::tensor alanı değerlerle doldurulur, diğer alanlar (harita ve sekans) tanımlanmaz.

Üç OnnxTypeInfo türünden (ONNX_TYPE_TENSOR, ONNX_TYPE_MAP veya ONNX_TYPE_SEQUENCE) yalnızca biri girdi olarak kullanılabilir. Türe bağlı olarak ilgili altyapı (OnnxTypeInfo::tensor, OnnxTypeInfo::map veya OnnxTypeInfo::sequence) doldurulur.

OnnxTensorTypeInfo

Yapı, ONNX modelinin girdi veya çıktı parametresindeki tensörü tanımlar.

```
struct OnnxTensorTypeInfo
 {
  const ENUM_ONNX_DATA_TYPE  data_type;      // tensördeki veri türü
  const long                 dimensions[];   // tensördeki eleman sayısı
 };
```

OnnxMapTypeInfo

Yapı, ONNX modelinin çıktı parametresinde elde edilen haritayı tanımlar.

```
struct OnnxMapTypeInfo
 {
  const ENUM_ONNX_DATA_TYPE   key_type;      // anahtar türü
  const OnnxTypeInfo&         value_type;    // değer türü
 };
```

OnnxSequenceTypeInfo

Yapı, ONNX modelinin çıktı parametresinde elde edilen sekansı tanımlar.

```
struct OnnxSequenceTypeInfo
 {
```

```
  const OnnxTypeInfo&          value_type;      // sekanstaki veri türü
};
```

ENUM_ONNX_TYPE

ENUM_ONNX_TYPE numaralandırması, model parametresinin türünü tanımlar.

Kimlik	Açıklama
ONNX_TYPE_UNKNOWN	Bilinmeyen
ONNX_TYPE_TENSOR	Tensör
ONNX_TYPE_SEQUENCE	Sekans
ONNX_TYPE_MAP	Harita
ONNX_TYPE_OPAQUE	Soyut (opak)
ONNX_TYPE_SPARSETENSOR	Seyrek tensör

ENUM_ONNX_DATA_TYPE

ENUM_ONNX_DATA_TYPE numaralandırması, kullanılan veri türünü tanımlar.

Kimlik	Açıklama
ONNX_DATA_TYPE_UNDEFINED	Tanımsız
ONNX_DATA_TYPE_FLOAT	float
ONNX_DATA_TYPE_INT8	8-bit int
ONNX_DATA_TYPE_UINT16	16-bit uint
ONNX_DATA_TYPE_INT16	16-bit int
ONNX_DATA_TYPE_INT32	32-bit int
ONNX_DATA_TYPE_INT64	64-bit int
ONNX_DATA_TYPE_STRING	string
ONNX_DATA_TYPE_BOOL	bool
ONNX_DATA_TYPE_FLOAT16	16-bit float
ONNX_DATA_TYPE_DOUBLE	double
ONNX_DATA_TYPE_UINT32	32-bit uint
ONNX_DATA_TYPE_UINT64	64-bit uint
ONNX_DATA_TYPE_COMPLEX64	64-bit complex

Kimlik	Açıklama
ONNX_DATA_TYPE_COMPLEX128	128-bit complex
ONNX_DATA_TYPE_BFLOAT16	16-bit bfloat (Brain Floating Point)

ENUM_ONNX_FLAGS

ENUM_ONNX_FLAGS numaralandırması, modelin çalışma modunu tanımlar.

Kimlik	Açıklama
ONNX_DEBUG_LOGS	Hata ayıklama günlüklerini yazdırır
ONNX_NO_CONVERSION	Otomatik dönüştürmeyi devre dışı bırakır, kullanıcı verilerini olduğu gibi kullanır
ONNX_COMMON_FOLDER	Common\Files klasöründen model dosyası yükler; değer, FILE_COMMON bayrağına eşittir

Standart Kütüphane

Bu bölümler MQL5 Standart Kütüphanesi için teknik detayları ve bileşenlerinin tariflerini içermektedir.

MQL5 Standart Kütüphanesi MQL5 ile yazılmıştır ve programların (göstergeler, scriptler ve uzmanlar) yazımı için tasarlanmıştır. Kütüphane birçok MQL5 içsel fonksiyonuna, uygun erişim sağlar.

MQL5 Standart Kütüphanesi terminalin çalışma konumunda 'Include' klasöründe yer alır.

Bölüm	Konum
Mathematics	Include\Math\
OpenCL	Include\OpenCL\
Temel Sınıf CObject	Include\
Veri Toplama	Include\Arrays\
Jenerik Data Koleksiyonları	Include\Generic\
Dosyalar	Include\Files\
Dizgiler	Include\Strings\
Grafik Nesneleri	Include\Objects\
Özel Grafikler	Include\Canvas\
3D grafik	Include\Canvas\
Fiyat Çizelgeleri	Include\Charts\
Scientific Charts	Include\Graphics\
Göstergeler	Include\Indicators\
Alım-satım sınıfları	Include\Trade\
Strateji Modülleri	Include\Expert\
Paneller ve İletişim Kutuları	Include\Controls\

Matematik

Çeşitli matematik alanlarında hesaplamalar gerçekleştirmek için bazı kütüphaneler hazırlanmıştır:

- Statistics - olasılık teorisinde yer alan çeşitli dağılımlarla çalışmak için tasarlanmış fonksiyonlar
- Bulanık mantık - Mamdani ve Sugeno bulanık çıkarım sistemleriyle çalışmak için tasarlanmış kütüphane
- ALGLIB - veri analizi (kümeleme, karar ağaçları, doğrusal regresyon, nöral ağlar), diferansiyel denklem çözme, Fourier dönüşümü, nümerik integrasyon, optimizasyon problemleri, istatistiksel analiz ve daha fazlası.

İstatistik

İstatistik kütüphanesi temel istatistiksel dağılımlarla çalışmak için tasarlanmış yöntemler içerir.

Kütüphanede her dağılım için beş adet fonksiyon bulunur:

1. Olasılık yoğunluğunun hesaplanması - MathProbabilityDensityX() biçimindeki fonksiyonlar
2. Olasılık hesabı - MathCumulativeDistributionX() biçimindeki fonksiyonlar
3. Kuantillerin hesaplanması - MathQuantileX() biçimindeki fonksiyonlar
4. Belirtilen dağılıma uygun rassal değişkenlerin oluşturulması - MathRandomX() biçimindeki fonksiyonlar
5. Dağılımların teorik momentlerinin hesaplanması - MathMomentsX() biçimindeki fonksiyonlar

Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

- İstatistiksel Karakteristikler
- Normal Dağılım
- Log-normal dağılım
- Beta dağılımı
- Merkez-dışı beta dağılımı
- Gamma dağılımı
- Ki-kare dağılımı
- Merkez-dışı ki-kare dağılımı
- Üssel dağılım
- F-dağılımı
- Merkez-dışı F-dağılımı
- t-dağılımı
- Merkez-dışı t-dağılımı
- Lojistik dağılım
- Cauchy dağılımı
- Tekdüze dağılım
- Weibull dağılımı
- Binomial dağılım
- Negatif binomial dağılım
- Geometrik dağılım
- Hipergeometrik dağılım
- Poisson dağılımı
- Alt Fonksiyonlar

Örnek:

```
//+------------------------------------------------------------------+
//|                                     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"
//--- normal dağılımı hesaplayan fonksiyonları dahil et
#include <Math\Stat\Normal.mqh>
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- normal dağılım için parametreleri ayarla
   double mu=5.0;
   double sigma=1.0;
   PrintFormat("mu=%G ve sigma=%G parametrelerine sahip normal dağılım, hesaplama örne
//--- aralığı ayarla
   double x1=mu-sigma;
   double x2=mu+sigma;
//--- olasılık hesabı için kullanılacak değişkenler
   double cdf1,cdf2,probability;
//--- hata kodu değişkenleri
   int error_code1,error_code2;
//--- dağılım fonksiyonlarının değerlerini hesapla
   cdf1=MathCumulativeDistributionNormal(x1,mu,sigma,error_code1);
   cdf2=MathCumulativeDistributionNormal(x2,mu,sigma,error_code2);
//--- hata kodlarını kontrol et
   if(error_code1==ERR_OK && error_code2==ERR_OK)
     {
      //--- belli aralıktaki rassal değişkenin olasılığını hesapla
      probability=cdf2-cdf1;
      //--- sonucu çıktıla
      PrintFormat("1. %.5f<x<%.5f aralığında, rassal değişkenin olasılığını hesapla",x
      PrintFormat("  Cevap: Olasılık = %5.8f",probability);
     }

//--- 95% güven aralığı için, rassal x değişkeninin yer aldığı aralığı hesapla
   probability=0.95; //  güven olasılığını hesapla
//--- aralık sınırlarındaki olasılıkları ayarla
   double p1=(1.0-probability)*0.5;
   double p2=probability+(1.0-probability)*0.5;
//--- aralık sınırlarını hesapla
   x1=MathQuantileNormal(p1,mu,sigma,error_code1);
   x2=MathQuantileNormal(p2,mu,sigma,error_code2);
//--- hata kodlarını kontrol et
   if(error_code1==ERR_OK && error_code2==ERR_OK)
     {
      //--- sonucu çıktıla
      PrintFormat("2. Güven düzeyi = %.2f için, rassal değişkenin aralığını hesapla",p
      PrintFormat("  Cevap: aralık  %5.8f <= x <=%5.8f",x1,x2);
     }

   PrintFormat("3. Dağılımın ilk dört momentinin teorik sayısal değerlerini hesapla");
//--- Bir rassal değişken dizisi oluştur, ilk dört momenti hesapla ve teorik değerleri
   int data_count=1000000;  // değerlerin sayısını ayarla ve diziyi hazırla
   double data[];
   ArrayResize(data,data_count);
//--- rassal değerler oluştur ve diziye doldur
```

```
   for(int i=0; i<data_count; i++)
     {
      data[i]=MathRandomNormal(mu,sigma,error_code1);
     }
//--- hesaplama için başlangıç indisini ve veri miktarını ayarla
   int start=0;
   int count=data_count;
//--- oluşturulan değerlerin ilk dört momentini hesapla
   double mean=MathMean(data,start,count);
   double variance=MathVariance(data,start,count);
   double skewness=MathSkewness(data,start,count);
   double kurtosis=MathKurtosis(data,start,count);
//--- teorik momentler için kullanılacak değişkenler
   double normal_mean=0;
   double normal_variance=0;
   double normal_skewness=0;
   double normal_kurtosis=0;
//--- hesaplanan momentlerin değerlerini göster
   PrintFormat("            Ortalama           Varyans          Çarpıklık           Ba
   PrintFormat("Hesaplanan  %.10f   %.10f      %.10f      %.10f",mean,variance,skewnes
//--- ilk dört momentin teorik değerlerini hesapla ve elde edilen değerlerle karşılaşt
   if(MathMomentsNormal(mu,sigma,normal_mean,normal_variance,normal_skewness,normal_ku
     {
      PrintFormat("Teorik %.10f   %.10f      %.10f       %.10f",normal_mean,normal_var
      PrintFormat("Fark   %.10f   %.10f      %.10f       %.10f",mean-normal_mean,variar
     }
  }
```

İstatistiksel Karakteristikler

Bu fonksiyon grubu, verilen dizinin istatistiksel karakteristiklerini hesaplar:

- ortalama,
- varyans,
- çarpıklık,
- basıklık,
- medyan,
- kök-ortalama-kare ve
- standart sapma.

Fonksiyon	Açıklama
MathMean	Dizi elemanlarının ortalamasını (ilk momentini) hesaplar
MathVariance	Dizi elemanlarının varyansını (ikinci momentini) hesaplar
MathSkewness	Dizi elemanlarının çarpıklığını (üçüncü momentini) hesaplar
MathKurtosis	Dizi elemanlarının basıklığını (dördüncü momentini) hesaplar
MathMoments	Dizi elemanlarının ilk dört momentini (ortalama, vayans, çarpıklık, basıklık) hesaplar
MathMedian	Dizi elemanlarının medyan değerini hesaplar
MathStandardDeviation	Dizi elemanlarının standart sapmasını hesaplar
MathAverageDeviation	Dizi elemanlarının mutlak sapmalarının ortalamasını hesaplar

MathMean

Dizi elemanlarının ortalamasını (ilk momentini) hesaplar. R dilindeki mean() fonksiyonunun analoğu.

```
double  MathMean(
   const double&  array[]                // veri dizisi
   );
```

Parametreler

array

[in] Ortalamasının hesaplanacağı verileri içeren dizi.

start=0

[in] Hesaplama için başlangıç indisi.

count=WHOLE_ARRAY

[in] hesaplanacak eleman sayısı.

Dönüş Değeri

Dizi elemanlarının ortalaması. Hata durumunda NaN dönüşü yapar.

MathVariance

Dizi elemanlarının varyansını (ikinci momentini) hesaplar. R dilindekivar() fonksiyonunun analoğu.

```
double  MathVariance(
   const double&  array[]                // veri dizisi
   );
```

Parametreler

array

[in] Hesaplanacak verileri içeren dizi.

start=0

[in] Hesaplama için başlangıç indisi.

count=WHOLE_ARRAY

[in] hesaplanacak eleman sayısı.

Dönüş Değeri

Dizi elemanlarının varyansı. Hata durumunda NaN dönüşü yapar.

MathSkewness

Dizi elemanlarının çarpıklığını (üçüncü momentini) hesaplar. R dilindekiskewness() in R (e1071 library).

```
double  MathSkewness(
   const double&  array[]                // veri dizisi
   );
```

Parametreler

array

[in] Hesaplanacak verileri içeren dizi.

start=0

[in] Hesaplama için başlangıç indisi.

count=WHOLE_ARRAY

[in] hesaplanacak eleman sayısı.

Dönüş Değeri

Dizi elemanlarının çarpıklığı. Hata durumunda NaN dönüşü yapar.

MathKurtosis

Dizi elemanlarının basıklığını (dördüncü momentini) hesaplar. R dilindeki (e1071 kütüphanesinin) kurtosis() fonksiyonunun analoğu.

```
double  MathKurtosis(
   const double&  array[]                // veri dizisi
   );
```

Parametreler

array

[in] Hesaplanacak verileri içeren dizi.

start=0

[in] Hesaplama için başlangıç indisi.

count=WHOLE_ARRAY

[in] hesaplanacak eleman sayısı.

Dönüş Değeri

Dizi elemanlarının basıklığı. Hata durumunda NaN dönüşü yapar.

Bilgilendirme

Basıklık hesabı, normal dağılımın aşırı basıklığı kullanılarak gerçekleştirilir (aşırı basıklık=basıklık-3), yani normal dağılımın aşırı basıklık değeri sıfırdır.

Beklenen değer üzerindeki zirve keskinse basıklık pozitif, değilse negatiftir.

MathMoments

Dizi elemanlarının ilk dört momentini (ortalama, vayans, çarpıklık, basıklık) hesaplar.

```
double  MathMoments(
   const double&  array[],              // veri dizisi
   double&        mean,                 // ortalama (1. moment)
   double&        variance,             // varyans (2. moment)
   double&        skewness,             // çarpıklık (3. moment)
   double&        kurtosis,             // basıklık (4. moment)
   const int      start=0,              // başlangıç indisi
   const int      count=WHOLE_ARRAY     // eleman sayısı
   );
```

Parametreler

array

[in] Hesaplanacak verileri içeren dizi.

mean

[out] Ortalama (1. moment) değişkeni

variance

[out] Varyans (2. moment) değişkeni

skewness

[out] Çarpıklık (3. moment) değişkeni

kurtosis

[out] Basıklık (4. moment) değişkeni

start=0

[in] Hesaplama için başlangıç indisi.

count=WHOLE_ARRAY

[in] hesaplanacak eleman sayısı.

Dönüş Değeri

Momentler başarıyla hesaplanırsa 'true', aksi durumda 'false' dönüşü yapar.

Bilgilendirme

Basıklık hesabı, normal dağılımın aşırı basıklığı kullanılarak gerçekleştirilir (aşırı basıklık=basıklık-3), yani normal dağılımın aşırı basıklık değeri sıfırdır.

Beklenen değer üzerindeki zirve keskinse basıklık pozitif, değilse negatiftir.

MathMedian

Dizi elemanlarının medyan değerini hesaplar. R dilindekimedian() fonksiyonunun analoğu.

```
double  MathMedian(
   const double&  array[]                // veri dizisi
   );
```

Parametreler

array

[in] Hesaplanacak verileri içeren dizi.

start=0

[in] Hesaplama için başlangıç indisi.

count=WHOLE_ARRAY

[in] hesaplanacak eleman sayısı.

Dönüş Değeri

Dizi elemanlarının medyan değeri. Hata durumunda NaN dönüşü yapar.

MathStandardDeviation

Dizi elemanlarının standart sapmasını hesaplar. R dilindekisd() fonksiyonunun analoğu.

```
double  MathStandardDeviation(
   const double&  array[]                // veri dizisi
   );
```

Parametreler

array

[in] Hesaplanacak verileri içeren dizi.

start=0

[in] Hesaplama için başlangıç indisi.

count=WHOLE_ARRAY

[in] hesaplanacak eleman sayısı.

Dönüş Değeri

Dizi elemanlarının standart sapması. Hata durumunda NaN dönüşü yapar.

MathAverageDeviation

Dizi elemanlarının mutlak sapmalarının ortalamasını hesaplar. R dilindeki aad() fonksiyonunun analoğu.

```
double  MathAverageDeviation(
   const double&  array[]                // veri dizisi
   );
```

Parametreler

array

[in] Hesaplanacak verileri içeren dizi.

start=0

[in] Hesaplama için başlangıç indisi.

count=WHOLE_ARRAY

[in] hesaplanacak eleman sayısı.

Dönüş Değeri

Dizi elemanlarının mutlak sapmalarının ortalaması. Hata durumunda NaN dönüşü yapar.

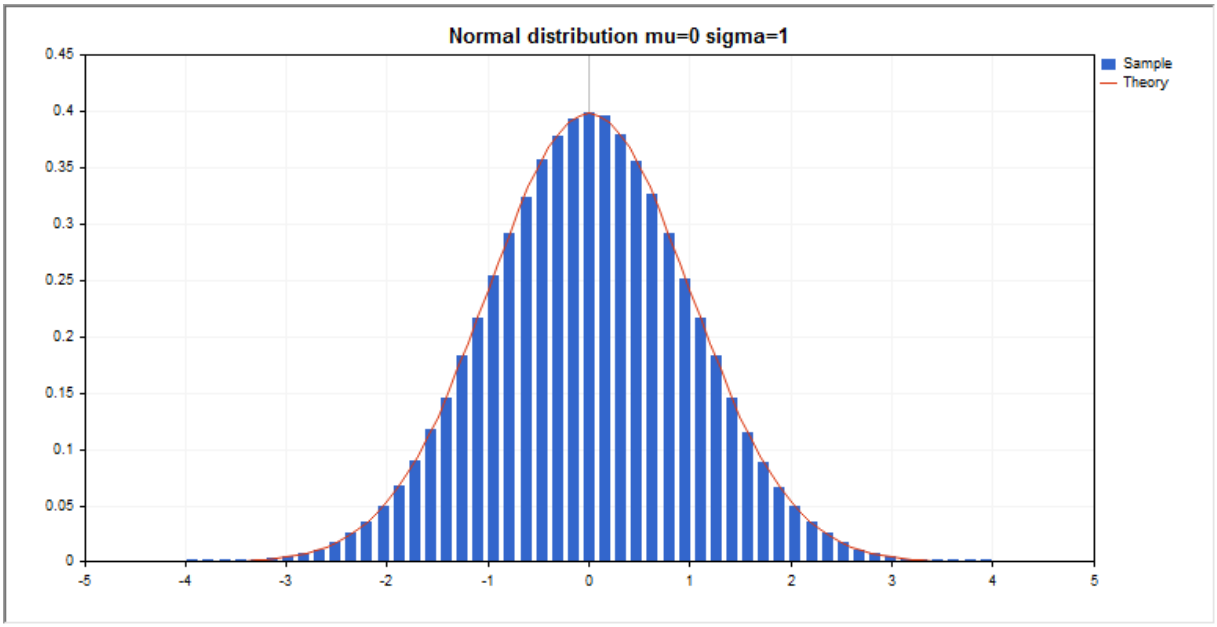
Normal Dağılım

Bu bölüm normal dağılım ile çalışmak için tasarlanmış fonksiyonlar içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. dağılım şu formülle tanımlanır:

$$f_{Normal}(x|\mu,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

burada:

- x — rassal değişkenin değeri
- μ — beklenen değer
- σ — kök-ortalama-kare sapması



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityNormal	Normal dağılımın olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionNormal	Normal dağılımın olasılık fonksiyonunun değerini hesaplar
MathQuantileNormal	Belli bir olasılık değeri için ters normal dağılımın olasılık fonksiyonunun değerini hesaplar
MathRandomNormal	Normal dağılım yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur

Fonksiyon	Açıklama
MathMomentsNormal	Normal dağılımın ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Normal.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double mean_value=0;  // beklenen değer
input double std_dev=1;     // standart sapma
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // örnekteki değerlerin sayısı
   int ncells=51;       // histogramdaki aralık sayısı
   double x[];          // histogram aralıklarının merkezleri
   double y[];          // aralığın içine düşen değerlerin sayısı
   double data[];       // rassal değişken örneği
   double max,min;      // örnekteki maksimum ve minimum değerlerin sayısı
//--- normal dağılıma uyan bir örnek oluştur
   MathRandomNormal(mean_value,std_dev,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNormal(x2,mean_value,std_dev,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
//--- tüm eğrileri çiz
   graphic.CurvePlotAll();
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

```
//+------------------------------------------------------------------+
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityNormal

Bir rassal x değişkeni için normal dağılımın olasılık yoğunluk fonksiyonunun değerini mu ve sigma parametreline göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityNormal(
   const double  x,              // rassal değişkenin değeri
   const double  mu,             // dağılımın ortalama parametresi (beklenen değer)
   const double  sigma,          // dağılımın sigma parametresi (kök-ortalama-kare sap
   const bool    log_mode,       // değerin logaritmasını hesapla
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için normal dağılımın olasılık yoğunluk fonksiyonunun değerini mu ve sigma parametreline göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityNormal(
   const double  x,              // rassal değişkenin değeri
   const double  mu,             // dağılımın ortalama parametresi (beklenen değer)
   const double  sigma,          // dağılımın sigma parametresi (kök-ortalama-kare sap
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için normal dağılımın olasılık yoğunluk fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidnorm() fonksiyonunun analoğu

```
bool  MathProbabilityDensityNormal(
   const double&  x[],           // rassal değişkenin değerlerini içeren dizi
   const double   mu,            // dağılımın ortalama parametresi (beklenen değer)
   const double   sigma,         // dağılımın sigma parametresi (kök-ortalama-kare sa
   const bool     log_mode,      // değerin logaritmasını hesapla
   double&        result[]       // olasılık yoğunluk fonksiyonunun değerlerini içere
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için normal dağılımın olasılık yoğunluk fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityNormal(
   const double&  x[],           // rassal değişkenin değerlerini içeren dizi
   const double   mu,            // dağılımın ortalama parametresi (beklenen değer)
   const double   sigma,         // dağılımın sigma parametresi (kök-ortalama-kare sa
   double&        result[]       // olasılık yoğunluk fonksiyonunun değerlerini içere
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

mu

[in] — dağılımın ortalama parametresi (beklenen değer)

sigma

[in] dağılımın sigma parametresi (kök-ortalama-kare sapması).

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerlerini alacak olan dizi.

MathCumulativeDistributionNormal

Bir rassal x değişkeni için normal dağılımın olasılık fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionNormal(
   const double  x,              // rassal değişkenin değeri
   const double  mu,             // beklenen değer
   const double  sigma,          // kök-ortalama-kare sapması
   const bool    tail,           // kuyruk için hesaplama bayrağı
   const bool    log_mode,       // değerin logaritmasını hesapla
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için normal dağılımın olasılık fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionNormal(
   const double  x,              // rassal değişkenin değeri
   const double  mu,             // beklenen değer
   const double  sigma,          // kök-ortalama-kare sapması
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için normal dağılımın olasılık fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidnorm() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionNormal(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  mu,             // beklenen değer
   const double  sigma,          // kök-ortalama-kare sapması
   const bool    tail,           // kuyruk için hesaplama bayrağı
   const bool    log_mode,       // değerin logaritmasını hesapla
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için normal dağılımın olasılık fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionNormal(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  mu,             // beklenen değer
   const double  sigma,          // kök-ortalama-kare sapması
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

mu

[in] — dağılımın ortalama parametresi (beklenen değer)

sigma

[in] dağılımın sigma parametresi (kök-ortalama-kare sapması).

tail

[in] hesaplama bayrağı. 'tail=true' ise, olasılık rassal değişkenin x'i aşmayan değerleri için hesaplanır.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Olasılık fonksiyonunun değerlerini alacak olan dizi.

MathQuantileNormal

Belirtilen *olasılık* değeri için, ters normal dağılımın olasılık fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileNormal(
   const double  probability,    // rassal değişkenin olasılığı
   const double  mu,             // beklenen değer
   const double  sigma,          // kök-ortalama-kare sapması
   const bool    tail,           // kuyruk için hesaplama bayrağı
   const bool    log_mode,       // değerin logaritmasını hesapla
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters normal dağılımın olasılık fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileNormal(
   const double  probability,    // rassal değişkenin olasılığı
   const double  mu,             // beklenen değer
   const double  sigma,          // kök-ortalama-kare sapması
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters normal dağılımın olasılık fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qnorm() fonksiyonunun analoğu

```
bool  MathQuantileNormal(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  mu,             // beklenen değer
   const double  sigma,          // kök-ortalama-kare sapması
   const bool    tail,           // kuyruk için hesaplama bayrağı
   const bool    log_mode,       // değerin logaritmasını hesapla
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters normal dağılımın olasılık fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileNormal(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  mu,             // beklenen değer
   const double  sigma,          // kök-ortalama-kare sapması
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

mu

[in] — dağılımın ortalama parametresi (beklenen değer)

sigma

[in] dağılımın sigma parametresi (kök-ortalama-kare sapması).

tail

[in] hesaplama bayrağı. 'false' ise hesaplama 1.0 olasılık için yapılır.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantillerin yazılacağı dizi.

MathRandomNormal

mu ve sigma parametrelerini kullanarak, normal dağılım yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomNormal(
   const double  mu,              // beklenen değer
   const double  sigma,           // kök-ortalama-kare sapması
   int&          error_code       // hata kodu değişkeni
   );
```

mu ve sigma parametrelerini kullanarak, normal dağılım yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirnorm() fonksiyonunun analoğu

```
bool  MathRandomNormal(
   const double  mu,              // beklenen değer
   const double  sigma,           // kök-ortalama-kare sapması
   const int     data_count,      // istenen veri miktarı
   double&       result[]         // pseudo-rassal değişkenleri içeren dizi
   );
```

Parametreler

mu

[in] — dağılımın ortalama parametresi (beklenen değer)

sigma

[in] dağılımın sigma parametresi (kök-ortalama-kare sapması).

data_count

[in] Alınan pseudo-rassal değişkenlerin sayısı.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsNormal

Normal dağılımın ilk dört momentinin teorik sayısal değerlerini hesaplar.

```
double  MathMomentsNormal(
   const double  mu,              // beklenen değer
   const double  sigma,           // kök-ortalama-kare sapması
   double&       mean,            // ortalama değişkeni
   double&       variance,        // varyans değişkeni
   double&       skewness,        // çarpıklık değişkeni
   double&       kurtosis,        // basıklık değişkeni
   int&          error_code       // hata kodu değişkeni
   );
```

Parametreler

mu

[in] — dağılımın ortalama parametresi (beklenen değer)

sigma

[in] dağılımın sigma parametresi (kök-ortalama-kare sapması).

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanırsa 'true', aksi durumda 'false' dönüşü yapar.

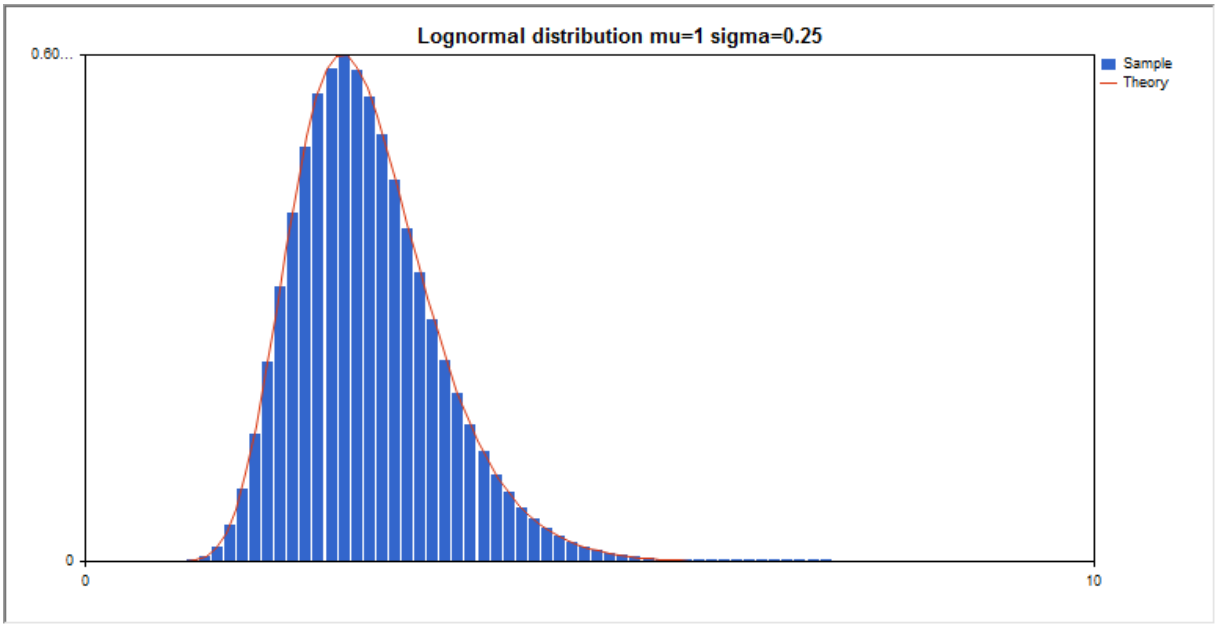
Log-normal dağılım

Bu bölüm log-normal dağılım ile çalışmak için tasarlanmış fonksiyonlar içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Log-normal dağılım şu formülle tanımlanır:

$$f_{Lognormal}(x|\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}$$

burada:

- x — rassal değişkenin değeri
- μ — beklenen değerin logaritması
- σ — kök-ortalama-kare sapmasının logaritması



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityLognormal	Log-normal dağılımın olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionLognormal	Log-normal dağılımın olasılık fonksiyonunun değerini hesaplar
MathQuantileLognormal	Belli bir olasılık değeri için ters log-normal dağılımın olasılık fonksiyonunun değerini hesaplar
MathRandomLognormal	Log-normal dağılım yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur
MathMomentsLognormal	Log-normal dağılımın ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Lognormal.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double mean_value=1.0;  // beklenen değerin logaritması
input double std_dev=0.25;    // standart sapmanın logaritması
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // örnekteki değerlerin sayısı
   int ncells=51;       // histogramdaki aralık sayısı
   double x[];          // histogram aralıklarının merkezleri
   double y[];          // aralığın içine düşen değerlerin sayısı
   double data[];       // rassal değişken örneği
   double max,min;      // örnekteki maksimum ve minimum değerlerin sayısı
//--- log normal dağılıma uyan bir örnek oluştur
   MathRandomLognormal(mean_value,std_dev,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityLognormal(x2,mean_value,std_dev,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- Y ekseninin otomatik olarak ölçeklenmesini engelle
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(theor_max);
   graphic.YAxis().Min(0);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

```
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityLognormal

Bir rassal x değişkeni için log-normal dağılımın olasılık yoğunluk fonksiyonunun değerini mu ve sigma parametreline göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityLognormal(
   const double  x,              // rassal değişkenin değeri
   const double  mu,             // beklenen değerin logaritması (log ortalama)
   const double  sigma,          // kök-ortalama-kare değerinin logaritması (log stand
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için log-normal dağılımın olasılık yoğunluk fonksiyonunun değerini mu ve sigma parametreline göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityLognormal(
   const double  x,              // rassal değişkenin değeri
   const double  mu,             // beklenen değerin logaritması (log ortalama)
   const double  sigma,          // kök-ortalama-kare değerinin logaritması (log stand
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için log-normal dağılımın olasılık yoğunluk fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar. R dilindeki dlnorm() fonksiyonunun analoğu

```
bool  MathProbabilityDensityLognormal(
   const double&  x[],           // rassal değişkenin değerlerini içeren dizi
   const double   mu,            // beklenen değerin logaritması (log ortalama)
   const double   sigma,         // kök-ortalama-kare değerinin logaritması (log star
   const bool     log_mode,      // değerin logaritmasını hesapla, log_mode=true ise
   double&        result[]       // olasılık yoğunluk fonksiyonunun değerlerini içere
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için log-normal dağılımın olasılık yoğunluk fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityLognormal(
   const double&  x[],           // rassal değişkenin değerlerini içeren dizi
   const double   mu,            // beklenen değerin logaritması (log ortalama)
   const double   sigma,         // kök-ortalama-kare değerinin logaritması (log star
   double&        result[]       // olasılık yoğunluk fonksiyonunun değerlerini içere
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

mu

[in] Beklenen değerin logaritması (log_ortalama).

sigma

[in] Kök-ortalama-kare-sapma değerinin logaritması (log standart sapma).

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerlerini alacak olan dizi.

MathCumulativeDistributionLognormal

Bir rassal x değişkeni için log-normal dağılımın olasılık fonksiyonunu mu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionLognormal(
   const double  x,              // rassal değişkenin değeri
   const double  mu,             // beklenen değerin logaritması (log ortalama)
   const double  sigma,          // kök-ortalama-kare değerinin logaritması (log stand
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal d
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise,
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için log-normal dağılımın olasılık fonksiyonunu mu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionLognormal(
   const double  x,              // rassal değişkenin değeri
   const double  mu,             // beklenen değerin logaritması (log ortalama)
   const double  sigma,          // kök-ortalama-kare değerinin logaritması (log stand
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için log-normal dağılımın olasılık fonksiyonunu mu ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekiplnorm() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionLognormal(
   const double&  x[],           // rassal değişkenin değerlerini içeren dizi
   const double   mu,            // beklenen değerin logaritması (log ortalama)
   const double   sigma,         // kök-ortalama-kare değerinin logaritması (log star
   const bool     tail,          // hesplama bayrağı, 'true' ise, x'i aşmayan rassal
   const bool     log_mode,      // değerin logaritmasını hesapla. log_mode=true ise
   double&        result[]       //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için mu ve sigma parametreleri ile log-normal dağılımın olasılık fonksiyonunu hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionLognormal(
   const double&  x[],           // rassal değişkenin değerlerini içeren dizi
   const double   mu,            // beklenen değerin logaritması (log ortalama)
   const double   sigma,         // kök-ortalama-kare değerinin logaritması (log star
   double&        result[]       //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

mu

[in] Beklenen değerin logaritması (log_ortalama).

sigma

[in] Kök-ortalama-kare-sapma değerinin logaritması (log standart sapma).

tail

[in] Hesaplama gecikmesi. 'true' ise x değerini aşmayan rassal değişkenlerin olasılığı hesaplanır

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerini alacak olan dizi.

MathQuantileLognormal

Belirtilen olasılık değeri için, ters log-normal dağılımın olasılık fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileLognormal(
   const double  probability,     // rassal değişkenin gerçekleşme olasılığı
   const double  mu,              // beklenen değerin logaritması (log ortalama)
   const double  sigma,           // kök-ortalama-kare değerinin logaritması (log stand
   const bool    tail,            // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,        // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code       // hata kodu değişkeni
   );
```

Belirtilen olasılık değeri için, ters log-normal dağılımın olasılık fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileLognormal(
   const double  probability,     // rassal değişkenin gerçekleşme olasılığı
   const double  mu,              // beklenen değerin logaritması (log ortalama)
   const double  sigma,           // kök-ortalama-kare değerinin logaritması (log stand
   int&          error_code       // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan probability[] dizisi için, ters log-normal dağılımın olasılık fonksiyonunun değerini mu sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qlnorm() fonksiyonunun analoğu

```
bool  MathQuantileLognormal(
   const double&  probability[],  // rassal değişkenin olasılık değerlerini içeren diz
   const double   mu,             // beklenen değerin logaritması (log ortalama)
   const double   sigma,          // kök-ortalama-kare değerinin logaritması (log star
   const bool     tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olas
   const bool     log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&        result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan probability[] dizisi için, ters log-normal dağılımın olasılık fonksiyonunun değerini mu sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileLognormal(
   const double&  probability[],  // rassal değişkenin olasılık değerlerini içeren diz
   const double   mu,             // beklenen değerin logaritması (log ortalama)
   const double   sigma,          // kök-ortalama-kare değerinin logaritması (log star
   double&        result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin gerçekleşme olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

mu

[in] Beklenen değerin logaritması (log_ortalama).

sigma

[in] Kök-ortalama-kare-sapma değerinin logaritması (log standart sapma).

tail

[in] Hesaplama bayrağı. 'false' ise hesaplama 1.0 olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomLognormal

mu ve sigma parametrelerini kullanarak, log-normal dağılım yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomLognormal(
   const double  mu,              // beklenen değerin logaritması (log ortalama)
   const double  sigma,           // kök-ortalama-kare değerinin logaritması (log stand
   int&          error_code       // hata kodu değişkeni
   );
```

mu ve sigma parametrelerini kullanarak, log-normal dağılım yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirlnorm() fonksiyonunun analoğu.

```
double  MathRandomLognormal(
   const double  mu,              // beklenen değerin logaritması (log ortalama)
   const double  sigma,           // kök-ortalama-kare değerinin logaritması (log stand
   const int     data_count,      // istenen veri miktarı
   double&       result[]         // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

mu

[in] Beklenen değerin logaritması (log_ortalama).

sigma

[in] Kök-ortalama-kare-sapma değerinin logaritması (log standart sapma).

data_count

[in] İstenen veri miktarı.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Pseudo-rassal değişkenlerin dizisi.

MathMomentsLognormal

Log-normal dağılımın ilk dört momentinin teorik sayısal değerlerini hesaplar. Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

```
double  MathMomentsLognormal(
   const double  mu,              // beklenen değerin logaritması (log ortalama)
   const double  sigma,           // kök-ortalama-kare değerinin logaritması (log stand
   double&       mean,            // ortalama değişkeni
   double&       variance,        // varyans değişkeni
   double&       skewness,        // çarpıklık değişkenii
   double&       kurtosis,        // basıklık değişkeni
   int&          error_code       // hata kodu değişkeni
   );
```

Parametreler

mu

[in] Beklenen değerin logaritması (log_ortalama).

sigma

[in] Kök-ortalama-kare-sapma değerinin logaritması (log standart sapma).

mean

[in] Ortalama değişkeni.

variance

[out] Varyans değişkeni.

skewness

[out] Çarpıklık değişkeni.

kurtosis

[out] Basıklık değişkeni.

error code

[out] Hata kodu değişkeni.

Dönüş Değeri

Momentler başarıyla hesaplanırsa 'true', aksi durumda 'false' dönüşü yapar.

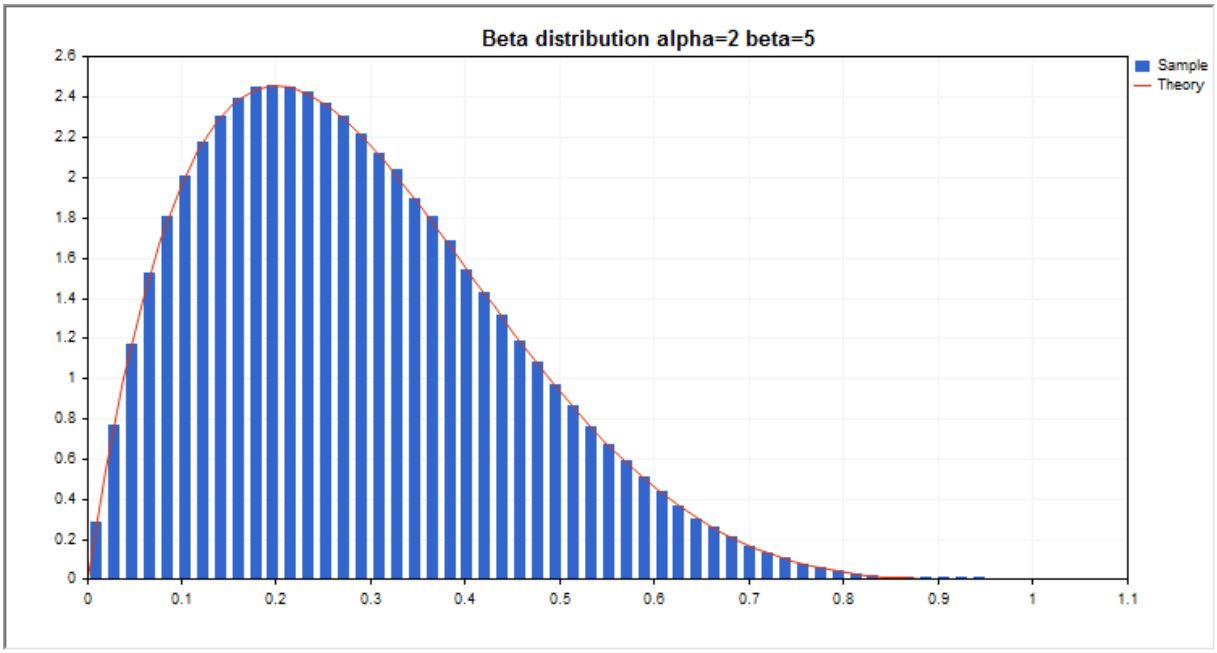
Beta dağılımı

Bu bölüm beta dağılımı ile çalışmak için tasarlanmış fonksiyonlar içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Beta dağılımı şu formülle tanımlanır:

$$f_{Beta}(x \mid a,b) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}$$

burada:

- x — rassal değişkenin değeri
- a — beta dağılımının ilk parametresi
- b — beta dağılımının ikinci parametresi



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityBeta	Beta dağılımının olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionBeta	Beta dağılımının olasılık fonksiyonunun değerini hesaplar
MathQuantileBeta	Belli bir olasılık değeri için ters beta dağılımının olasılık fonksiyonunun değerini hesaplar
MathRandomBeta	Beta dağılımı yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur
MathMomentsBeta	Beta dağılımının ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Beta.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double alpha=2;   // the beta dağılımının ilk parametresi (şekil 1)
input double beta=5;    // the beta dağılımının ikinci parametresi (şekil 2)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // örnekteki değerlerin sayısı
   int ncells=51;       // histogramdaki aralık sayısı
   double x[];          // histogram aralıklarının merkezleri
   double y[];          // aralığın içine düşen değerlerin sayısı
   double data[];       // rassal değişken örneği
   double max,min;      // örnekteki maksimum ve minimum değerlerin sayısı
//--- beta dağılımına uyan bir örnek oluştur
   MathRandomBeta(alpha,beta,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityBeta(x2,alpha,beta,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
```

```
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityBeta

Bir rassal x değişkeni için beta dağılımının olasılık yoğunluk fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityBeta(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise o
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için beta dağılımının olasılık yoğunluk fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityBeta(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için beta dağılımının olasılık yoğunluk fonksiyonununun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidbeta() fonksiyonunun analoğu

```
bool  MathProbabilityDensityBeta(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log_
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için beta dağılımının olasılık yoğunluk fonksiyonununun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityBeta(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

a

[in] Beta dağılımının ilk parametresi (şekil 1).

b

[in] Beta dağılımının ikinci parametresi (şekil 2)

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionBeta

Bir rassal x değişkeni için beta dağılımının olasılık fonksiyonunu a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionBeta(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal de
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise, d
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için beta dağılımının olasılık fonksiyonunu a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionBeta(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için beta dağılımının olasılık fonksiyonunu a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekipbeta() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionBeta(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal d
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise d
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

<t0>Rassal değişkenlerden oluşan bir x[] dizisi için beta dağılımının olasılık fonksiyonunu a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionBeta(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

a

[in] Beta dağılımının ilk parametresi (şekil 1).

b

[in] Beta dağılımının ikinci parametresi (şekil 2)

tail

[in] Hesaplama gecikmesi. 'true' ise x değerini aşmayan rassal değişkenlerin olasılığı hesaplanır

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerinin dizisi.

MathQuantileBeta

Belirtilen *olasılık* değeri için, ters beta dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileBeta(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters beta dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileBeta(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters beta dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qbeta() fonksiyonunun analoğu

```
double  MathQuantileBeta(
   const double&  probability[], // rassal değişkenin olasılık değerlerini içeren dizi
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters beta dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileBeta(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

a

[in] Beta dağılımının ilk parametresi (şekil 1).

b

[in] Beta dağılımının ikinci parametresi (şekil 2).

tail

[in] Hesaplama bayrağı. lower_tail=false ise hesaplama 1.0-olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomBeta

a ve b parametrelerini kullanarak, beta dağılımı yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomBeta(
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   int&          error_code      // hata kodu değişkeni
   );
```

a ve b parametrelerini kullanarak, beta dağılımı yasasına uygun olarak dağılmış bir pseudo-rassal değişken dizisi oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirbeta() fonksiyonunun analoğu

```
bool  MathRandomBeta(
   const double  a,               // beta dağılımının ilk parametresi (şekil1)
   const double  b,               // beta dağılımının ikinci parametresi (şekil2)
   const int     data_count,      // istenen veri miktarı
   double&       result[]         // pseudo-rassal değişkenleri içeren dizi
   );
```

Parametreler

a

[in] Beta dağılımının ilk parametresi (şekil 1).

b

[in] Beta dağılımının ikinci parametresi (şekil 2).

data_count

[in] Alınan pseudo-rassal değişkenlerin sayısı.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsBeta

Beta dağılımının ilk dört momentinin teorik sayısal değerlerini hesaplar.

```
double  MathMomentsBeta(
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   double&       mean,           // ortalama değişkeni
   double&       variance,       // varyans değişkeni
   double&       skewness,       // çarpıklık değişkeni
   double&       kurtosis,       // basıklık değişkeni
   int&          error_code      // hata kodu değişkeni
   );
```

Parametreler

a

[in] Beta dağılımının ilk parametresi (şekil 1).

b

[in] Beta dağılımının ikinci parametresi (şekil 2).

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanırsa 'true', aksi durumda 'false' dönüşü yapar.

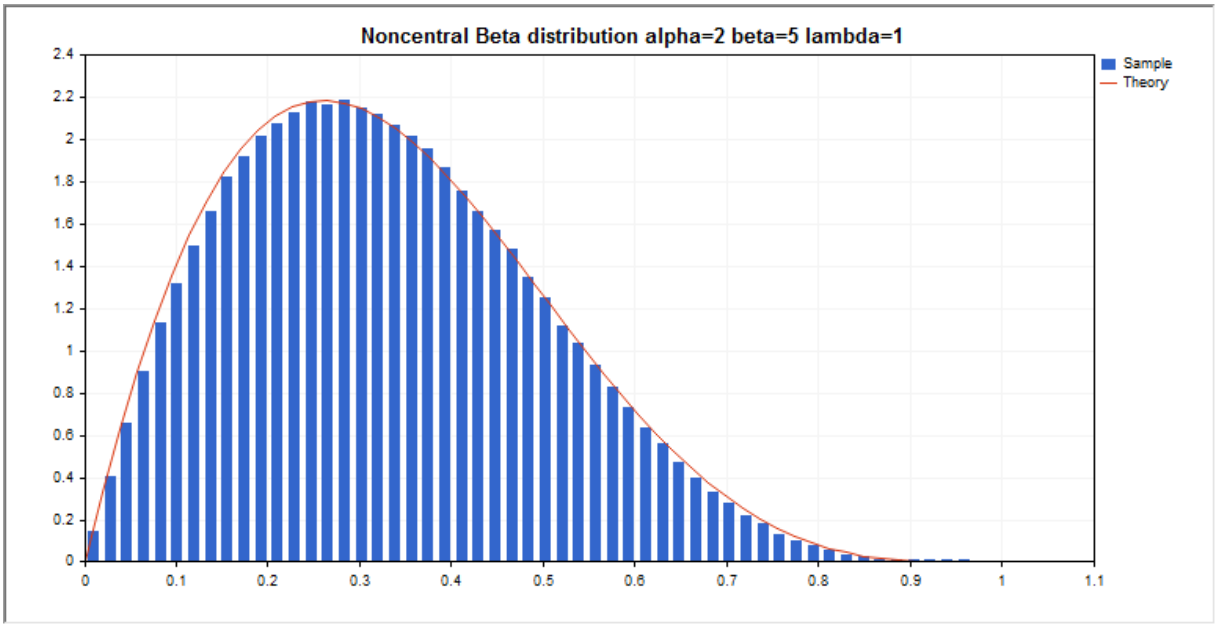
Merkez-dışı beta dağılımı

Bu bölüm merkez-dışı beta dağılımı ile çalışmak için gerekli olan fonksiyonları içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Merkez-dışı beta dağılımı şu formülle tanımlanır:

$$f_{NoncentralBeta}(x|a,b,\lambda)=\sum_{r=0}^{\infty}e^{-\frac{\lambda}{2}}\frac{\left(\frac{\lambda}{2}\right)^r}{r!}\frac{x^{a+r-1}(1-x)^{b-1}}{B(a+r,b)}$$

burada:

- x — rassal değişkenin değeri
- a — beta dağılımının ilk parametresi
- b — beta dağılımının ikinci parametresi
- λ — merkez-dışılık parametresi



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityNoncentralBeta	Merkez-dışı beta dağılımının olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionNoncentralBeta	Merkez-dışı beta dağılımının olasılık fonksiyonunun değerini hesaplar
MathQuantileNoncentralBeta	Ters merkez-dışı beta dağılımının olasılık fonksiyonunun değerini belirtilen olasılık değeri için hesaplar

Fonksiyon	Açıklama
MathRandomNoncentralBeta	Merkez-dışı beta dağılımı yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur
MathMomentsNoncentralBeta	Merkez-dışı beta dağılımının ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralBeta.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double a_par=2;     // beta dağılımının ilk parametresi (şekil 1)
input double b_par=5;     // the beta dağılımının ikinci parametresi (şekil 2)
input double l_par=1;     // merkezdışılık parametresi (lambda)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // örnekteki değerlerin sayısı
   int ncells=53;        // histogramdaki aralık sayısı
   double x[];           // histogram aralıklarının merkezleri
   double y[];           // aralığın içine düşen değerlerin sayısı
   double data[];        // rassal değişken örneği
   double max,min;       // örnekteki maksimum ve minimum değerlerin sayısı
//--- merkezdışı beta dağılımına uyan bir örnek oluştur
   MathRandomNoncentralBeta(a_par,b_par,l_par,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralBeta(x2,a_par,b_par,l_par,false,y2);
```

```
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
```

```
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityNoncentralBeta

Bir rassal x değişkeni için merkez-dışı beta dağılımının olasılık yoğunluk fonksiyonunun değerini a ve b ve lambda parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityNoncentralBeta(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const double  lambda,         // merkezdışılık parametresi
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise o
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için merkez-dışı beta dağılımının olasılık yoğunluk fonksiyonunun değerini a ve b ve lambda parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityNoncentralBeta(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const double  lambda,         // merkezdışılık parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için merkez-dışı beta dağılımının olasılık yoğunluk fonksiyonununun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidbeta() fonksiyonunun analoğu

```
bool  MathProbabilityDensityNoncentralBeta(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const double  lambda,         // merkezdışılık parametresi
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log_
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için merkez-dışı beta dağılımının olasılık yoğunluk fonksiyonununun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityNoncentralBeta(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const double  lambda,         // merkezdışılık parametresi
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

a

[in] Beta dağılımının ilk parametresi (şekil 1).

b

[in] Beta dağılımının ikinci parametresi (şekil 2)

lambda

[in] Merkezdışılık parametresi

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionNoncentralBeta

Bir rassal x değişkeni için merkez-dışı beta dağılımının olasılık fonksiyonunu a, b ve lambda parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionNoncentralBeta(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const double  lambda,         // merkezdışılık parametresi
   const bool    tail,           // hesaplama bayrağı. 'true' ise x değerini aşmayan ra
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise, c
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için merkez-dışı beta dağılımının olasılık fonksiyonunu a, b ve lambda parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionNoncentralBeta(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const double  lambda,         // merkezdışılık parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için merkez-dışı beta dağılımının olasılık fonksiyonunu a, b ve lambda parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekipbeta() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionNoncentralBeta(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const double  lambda,         // merkezdışılık parametresi
   const bool    tail,           // hesaplama bayrağı. 'true' ise x değerini aşmayan
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için merkez-dışı beta dağılımının olasılık fonksiyonunu a, b ve lambda parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionNoncentralBeta(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const double  lambda,         // merkezdışılık parametresi
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

a

[in] Beta dağılımının ilk parametresi (şekil 1).

b

[in] Beta dağılımının ikinci parametresi (şekil 2)

lambda

[in] Merkezdışılık parametresi

tail

[in] hesaplama bayrağı. 'true' ise, olasılık rassal değişkenin x'i aşmayan değerleri için hesaplanır.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerinin dizisi.

MathQuantileNoncentralBeta

Belirtilen *olasılık* değeri için, ters merkez-dışı beta dağılımınının olasılık fonksiyonunun değerini a, b ve lambda parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileNoncentralBeta(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const double  lambda,         // merkezdışılık parametresi
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters merkez-dışı beta dağılımınının olasılık fonksiyonunun değerini a, b ve lambda parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileNoncentralBeta(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const double  lambda,         // merkezdışılık parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan probability[] dizisi için, ters merkez-dışı beta dağılımının olasılık fonksiyonunun değerini a, b ve lambda parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekiqbeta() fonksiyonunun analoğu

```
double  MathQuantileNoncentralBeta(
   const double&  probability[],  // rassal değişkenin olasılık değerlerini içeren diz
   const double  a,               // beta dağılımının ilk parametresi (şekil1)
   const double  b,               // beta dağılımının ikinci parametresi (şekil2)
   const double  lambda,          // merkezdışılık parametresi
   const bool    tail,            // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,        // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]         // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan probability[] dizisi için, ters merkez-dışı beta dağılımının olasılık fonksiyonunun değerini a, b ve lambda parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileNoncentralBeta(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const double  lambda,         // merkezdışılık parametresi
   double&       result[]        // kuantil değerlerinin dizisi
```

```
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

a

[in] Beta dağılımının ilk parametresi (şekil 1).

b

[in] Beta dağılımının ikinci parametresi (şekil 2).

lambda

[in] Merkezdışılık parametresi.

tail

[in] Hesaplama bayrağı. 'false' ise hesaplama 1.0 olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomNoncentralBeta

a, b ve lambda parametrelerini kullanarak, merkez-dışı beta dağılımı yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomNoncentralBeta(
   const double  a,             // beta dağılımının ilk parametresi (şekil1)
   const double  b,             // beta dağılımının ikinci parametresi (şekil2)
   const double  lambda,        // merkezdışılık parametresi
   int&          error_code     // hata kodu değişkeni
   );
```

a, b ve lambda parametrelerini kullanarak, merkez-dışı beta dağılımı yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirbeta() fonksiyonunun analoğu

```
bool  MathRandomNoncentralBeta(
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const double  lambda,         // merkezdışılık parametresi
   const int     data_count,     // istenen veri miktarı
   double&       result[]        // pseudo-rassal değişkenleri içeren dizi
   );
```

Parametreler

a

[in] Beta dağılımının ilk parametresi (şekil 1).

b

[in] Beta dağılımının ikinci parametresi (şekil 2).

lambda

[in] Merkezdışılık parametresi

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsNoncentralBeta

Merkez-dışı beta dağılımının ilk dört momentinin teorik sayısal değerlerini a, b ve lambda parametrelerine göre hesaplar.

```
double  MathMomentsNoncentralBeta(
   const double  a,              // beta dağılımının ilk parametresi (şekil1)
   const double  b,              // beta dağılımının ikinci parametresi (şekil2)
   const double  lambda,         // merkezdışılık parametresi
   double&       mean,           // ortalama değişkeni
   double&       variance,       // varyans değişkeni
   double&       skewness,       // çarpıklık değişkeni
   double&       kurtosis,       // basıklık değişkeni
   int&          error_code      // hata kodu değişkeni
   );
```

Parametreler

a

[in] Beta dağılımının ilk parametresi (şekil 1).

b

[in] Beta dağılımının ikinci parametresi (şekil 2).

lambda

[in] Merkezdışılık parametresi

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

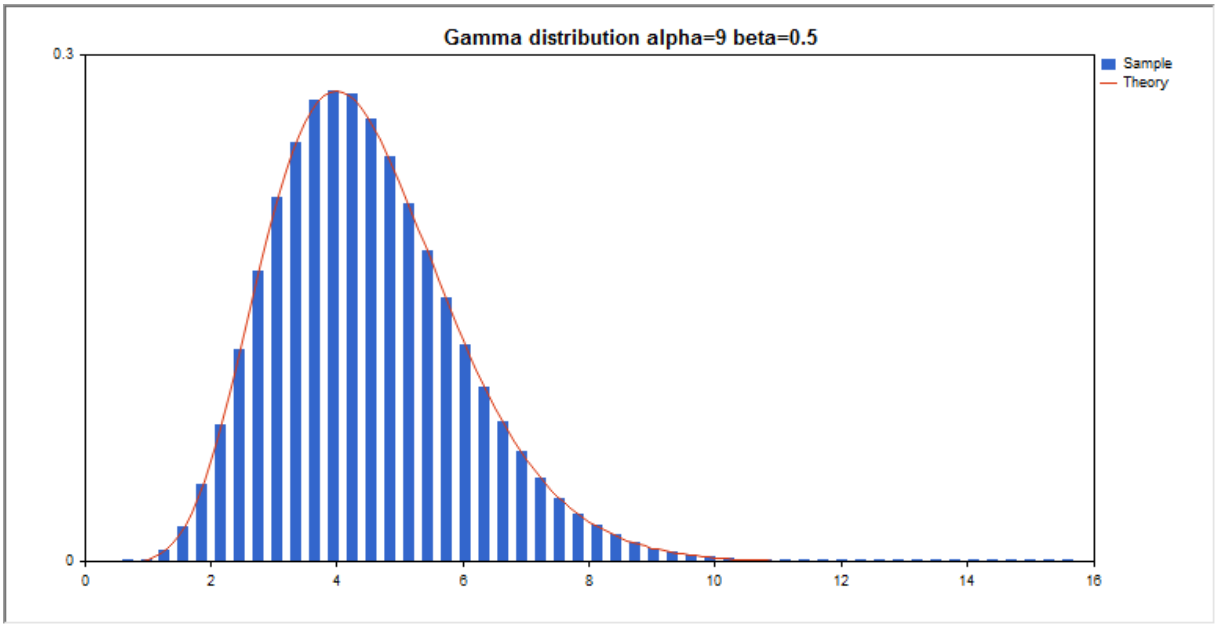
Gamma dağılımı

Bu bölüm gama dağılımı ile çalışmak için tasarlanan fonksiyonları içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Gamma dağılımı şu formülle tanımlanır:

$$f_{Gamma}(x|a,b) = \frac{1}{b^a \Gamma(a)} x^{a-1} e^{-\frac{x}{b}}$$

burada:

- x — rassal değişkenin değeri
- a — birinci dağolom parametresi
- b — ikinci dağılım parametresi



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityGamma	Gama dağılımının olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionGamma	Gama dağılımının olasılık fonksiyonunun değerini hesaplar
MathQuantileGamma	Belli bir olasılık değeri için ters gama dağılımının olasılık fonksiyonunun değerini hesaplar
MathRandomGamma	Gama dağılımı yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur
MathMomentsGamma	Gama dağılımının ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Gamma.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double alpha=9;   // dağılımın ilk parametresi (şekil)
input double beta=0.5;  // dağılımın ikinci parametresi (ölçek)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // örnekteki değerlerin sayısı
   int ncells=51;       // histogramdaki aralık sayısı
   double x[];          // histogram aralıklarının merkezleri
   double y[];          // aralığın içine düşen değerlerin sayısı
   double data[];       // rassal değişken örneği
   double max,min;      // örnekteki maksimum ve minimum değerlerin sayısı
//--- gama dağılımına uyan bir örnek oluştur
   MathRandomGamma(alpha,beta,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityGamma(x2,alpha,beta,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- Y ekseninin otomatik olarak ölçeklenmesini engelle
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(NormalizeDouble(theor_max,1));
   graphic.YAxis().Min(0);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

```
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityGamma

Bir rassal x değişkeni için gama dağılımının olasılık yoğunluk fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityGamma(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // dağılımın ilk parametresi (şekil)
   const double  b,              // dağılımın ikinci parametresi (ölçekl)
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise o
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için gama dağılımının olasılık yoğunluk fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityGamma(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // dağılımın ilk parametresi (şekil)
   const double  b,              // dağılımın ikinci parametresi (ölçekl)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için gama dağılımının olasılık yoğunluk fonksiyonununun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidgamma() fonksiyonunun analoğu

```
bool  MathProbabilityDensityGamma(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // dağılımın ilk parametresi (şekil)
   const double  b,              // dağılımın ikinci parametresi (ölçekl)
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için gama dağılımının olasılık yoğunluk fonksiyonununun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityGamma(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // dağılımın ilk parametresi (şekil)
   const double  b,              // dağılımın ikinci parametresi (ölçekl)
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

a

[in] Dağılımın ilk parametresi (şekil).

b

[in] Dağılımın ikinci parametresi (ölçek).

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionGamma

Bir rassal x değişkeni için gama dağılımının olasılık fonksiyonunu a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionGamma(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // dağılımın ilk parametresi (şekil)
   const double  b,              // dağılımın ikinci parametresi (ölçekl)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal de
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise, d
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için gama dağılımının olasılık fonksiyonunu a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionGamma(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // dağılımın ilk parametresi (şekil)
   const double  b,              // dağılımın ikinci parametresi (ölçekl)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, gama dağılımının olasılık fonksiyonunu a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekipgamma() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionGamma(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // dağılımın ilk parametresi (şekil)
   const double  b,              // dağılımın ikinci parametresi (ölçekl)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal d
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise d
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, gama dağılımının olasılık fonksiyonunu a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionGamma(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // dağılımın ilk parametresi (şekil)
   const double  b,              // dağılımın ikinci parametresi (ölçekl)
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

a

[in] Dağılımın ilk parametresi (şekil).

b

[in] Dağılımın ikinci parametresi (ölçek)

tail

[in] hesaplama bayrağı. 'true' ise, olasılık rassal değişkenin x'i aşmayan değerleri için hesaplanır.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerinin dizisi.

MathQuantileGamma

Belirtilen *olasılık* değeri için, ters gama dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileGamma(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  a,              // dağılımın ilk parametresi (şekil)
   const double  b,              // dağılımın ikinci parametresi (ölçekl)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters gama dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileGamma(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  a,              // dağılımın ilk parametresi (şekil)
   const double  b,              // dağılımın ikinci parametresi (ölçekl)
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters gama dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qgamma() fonksiyonunun analoğu

```
double  MathQuantileGamma(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  a,              // dağılımın ilk parametresi (şekil)
   const double  b,              // dağılımın ikinci parametresi (ölçekl)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters gama dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileGamma(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  a,              // dağılımın ilk parametresi (şekil)
   const double  b,              // dağılımın ikinci parametresi (ölçekl)
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

a

[in] Dağılımın ilk parametresi (şekil).

b

[in] Dağılımın ikinci parametresi (ölçek).

tail

[in] Hesaplama bayrağı. 'false' ise hesaplama 1.0 olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomGamma

a ve b parametrelerini kullanarak, gama dağılımı yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomGamma(
   const double  a,              // dağılımın ilk parametresi (şekil)
   const double  b,              // dağılımın ikinci parametresi (ölçekl)
   int&          error_code      // hata kodu değişkeni
   );
```

a ve b parametrelerini kullanarak, gama dağılımı yasasına uygun olarak dağılan bir pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirgamma() fonksiyonunun analoğu

```
bool  MathRandomGamma(
   const double  a,               // dağılımın ilk parametresi (şekil)
   const double  b,               // dağılımın ikinci parametresi (ölçekl)
   const int     data_count,      // istenen veri miktarı
   double&       result[]         // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

a

[in] Dağılımın ilk parametresi (şekil).

b

[in] Dağılımın ikinci parametresi (ölçek).

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsGamma

Gama dağılımının ilk dört momentinin teorik sayısal değerlerini a ve b parametrelerine göre hesaplar.

```
double  MathMomentsGamma(
   const double  a,              // dağılımın ilk parametresi (şekil)
   const double  b,              // dağılımın ikinci parametresi (ölçekl)
   double&       mean,           // ortalama değişkeni
   double&       variance,       // varyans değişkeni
   double&       skewness,       // çarpıklık değişkeni
   double&       kurtosis,       // basıklık değişkeni
   int&          error_code      // hata kodu değişkeni
   );
```

Parametreler

a

[in] Dağılımın ilk parametresi (şekil).

b

[in] Dağılımın ikinci parametresi (ölçek).

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

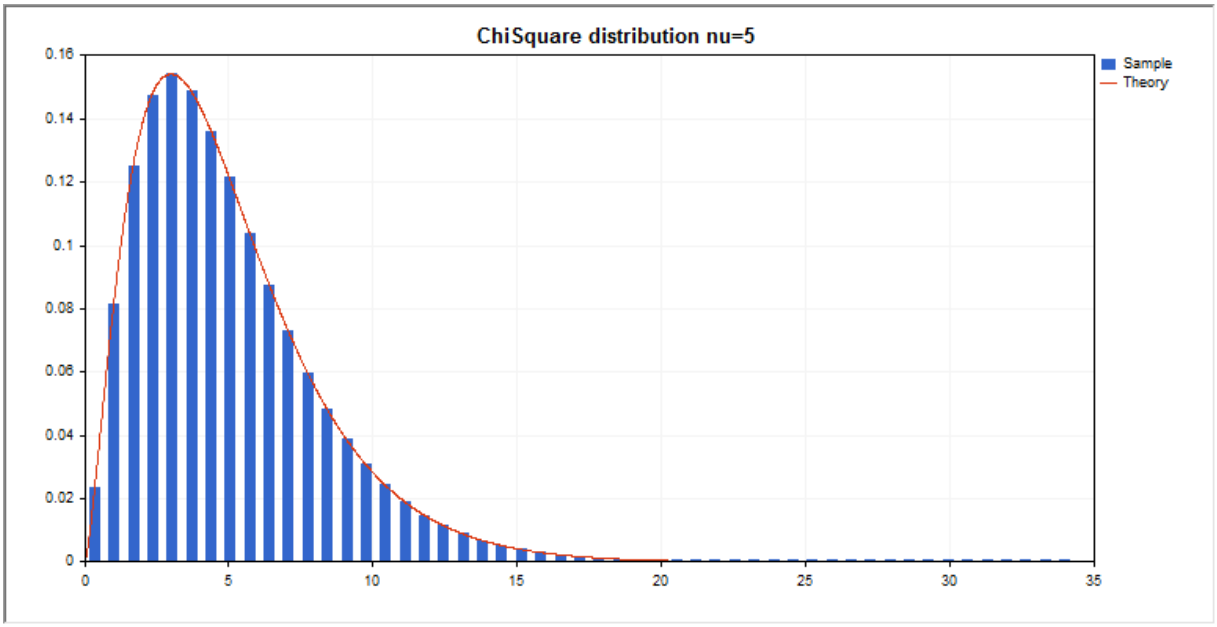
Ki-kare dağılımı

Bu bölüm ki-kare dağılımı ile çalışmak için tasarlanmış fonksiyonlar içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Ki-kare dağılımı şu formülle tanımlanır:

$$f_{Chi-Square}(x|\nu) = \frac{x^{\frac{(\nu-2)}{2}} e^{-\frac{x}{2}}}{2^{\frac{\nu}{2}}\Gamma\left(\frac{\nu}{2}\right)}$$

burada:

- x — rassal değişkenin değeri
- ν — derbestlik derecesi



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityChiSquare	Ki-kare dağılımının olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionChiSquare	Ki-kare dağılımının olasılık fonksiyonunun değerini hesaplar
MathQuantileChiSquare	Belli bir olasılık değeri için ters ki-kare dağılımının olasılık fonksiyonunun değerini hesaplar
MathRandomChiSquare	Ki-kare dağılımı yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur

Fonksiyon	Açıklama
MathMomentsChiSquare	Ki-kare dağılımının ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\ChiSquare.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double nu_par=5;    // serbestlik derecesi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // örnekteki değerlerin sayısı
   int ncells=51;       // histogramdaki aralık sayısı
   double x[];          // histogram aralıklarının merkezleri
   double y[];          // aralığın içine düşen değerlerin sayısı
   double data[];       // rassal değişken örneği
   double max,min;      // örnekteki maksimum ve minimum değerlerin sayısı
//--- ki-kare dağılımına uyan bir örnek oluştur
   MathRandomChiSquare(nu_par,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityChiSquare(x2,nu_par,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

```
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityChiSquare

Bir rassal x değişkeni için ki-kare dağılımının olasılık yoğunluk fonksiyonunun değerini nu parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityChiSquare(
   const double  x,              // rassal değişkenin değeri
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise o
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için ki-kare dağılımının olasılık yoğunluk fonksiyonunun değerini nu parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityChiSquare(
   const double  x,              // rassal değişkenin değeri
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için ki-kare dağılımının olasılık yoğunluk fonksiyonunun değerini nu parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidchisq() fonksiyonunun analoğu

```
bool  MathProbabilityDensityChiSquare(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için ki-kare dağılımının olasılık yoğunluk fonksiyonunun değerini nu parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityChiSquare(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

nu

[in] Dağılım parametresi (serbestlik derecesi)

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionChiSquare

Bir rassal x değişkeni için ki-kare dağılımının olasılık fonksiyonunu nu parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionChiSquare(
   const double  x,              // rassal değişkenin değeri
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal de
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise, d
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için ki-kare dağılımının olasılık fonksiyonunu nu parametresine göre hesaplar. Hata durumunda dönüş değeri: NaN

```
double  MathCumulativeDistributionChiSquare(
   const double  x,              // rassal değişkenin değeri
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için ki-kare dağılımının olasılık fonksiyonunu nu parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekipchisq() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionChiSquare(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal d
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise d
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için ki-kare dağılımının olasılık fonksiyonunu nu parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionChiSquare(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

nu

[in] Dağılım parametresi (serbestlik derecesi).

tail

[in] hesaplama bayrağı. 'true' ise, olasılık rassal değişkenin x'i aşmayan değerleri için hesaplanır.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerinin dizisi.

MathQuantileChiSquare

Belirtilen *olasılık* değeri için, ters ki-kare dağılımının olasılık fonksiyonunun değerini hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileChiSquare(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters ki-kare dağılımının olasılık fonksiyonunun değerini hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileChiSquare(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters ki-kare dağılımının olasılık fonksiyonunun değerini hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qchisq() fonksiyonunun analoğu

```
double  MathQuantileChiSquare(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters ki-kare dağılımının olasılık fonksiyonunun değerini hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileChiSquare(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

nu

[in] Dağılım parametresi (serbestlik derecesi).

tail

[in] Hesaplama bayrağı. 'false' ise hesaplama 1.0 olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomChiSquare

nu parametresini kullanarak, ki-kare dağılımı yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomChiSquare(
   const double  nu,            // dağılım parametresi (serbestlik derecesi)
   int&          error_code     // hata kodu değişkeni
   );
```

nu parametresini kullanarak, ki-kare dağılımı yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirchisq() fonksiyonunun analoğu

```
bool  MathRandomChiSquare(
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const int     data_count,     // istenen veri miktarı
   double&       result[]        // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

nu

[in] Dağılım parametresi (serbestlik derecesi).

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsChiSquare

Ki-kare dağılımının ilk dört momentinin teorik sayısal değerlerini nu parametresi ile hesaplar.

```
double  MathMomentsChiSquare(
   const double  nu,              // dağılım parametresi (serbestlik derecesi)
   double&       mean,            // ortalama değişkeni
   double&       variance,        // varyans değişkeni
   double&       skewness,        // çarpıklık değişkeni
   double&       kurtosis,        // basıklık değişkeni
   int&          error_code       // hata kodu değişkeni
   );
```

Parametreler

nu

[in] Dağılım parametresi (serbestlik derecesi).

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

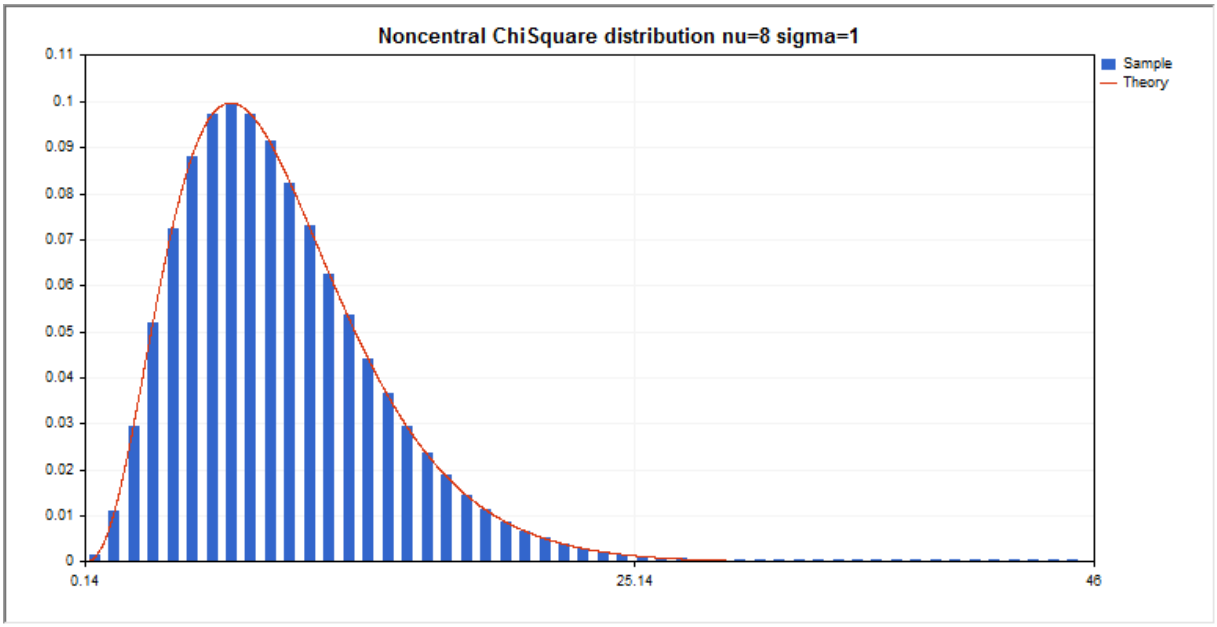
Merkez-dışı ki-kare dağılımı

Bu bölüm merkez-dışı ki-kare dağılımı ile çalışmak için tasarlanmış fonksiyonlar içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Merkez-dışı ki-kare dağılımı şu formülle tanımlanır:

$$f_{NoncentralChiSquare}(x|\nu,\sigma) = \frac{1}{2^{\frac{\nu}{2}}\Gamma\left(\frac{1}{2}\right)} x^{\frac{\nu}{2}-1} e^{-\frac{(x+\sigma)}{2}} \sum_{r=0}^{\infty} \frac{(\lambda x)^r}{(2r)!} \frac{\Gamma\left(\frac{1}{2}+r\right)}{\Gamma\left(\frac{\nu}{2}+r\right)}$$

burada:

- x — rassal değişkenin değeri
- ν — derbestlik derecesi
- σ — merkez-dışılık parametresi



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityNoncentralChiSquare	Merkez-dışı ki-kare dağılımının olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionNoncentralChiSquare	Merkez-dışı ki-kare dağılımının olasılık fonksiyonunun değerini hesaplar
MathQuantileNoncentralChiSquare	Belli bir olasılık değeri için ters merkez-dışı ki-kare dağılımının olasılık fonksiyonunun değerini hesaplar
MathRandomNoncentralChiSquare	Merkez-dışı ki-kare dağılımı yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur

Fonksiyon	Açıklama
MathMomentsNoncentralChiSquare	Merkez-dışı ki-kare dağılımının ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralChiSquare.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double nu_par=8;     // sebestlik derecesi
input double si_par=1;     // merkez-dışılık parametresi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // örnekteki değerlerin sayısı
   int ncells=51;        // histogramdaki aralık sayısı
   double x[];           // histogram aralıklarının merkezleri
   double y[];           // aralığın içine düşen değerlerin sayısı
   double data[];        // rassal değişken örneği
   double max,min;       // örnekteki maksimum ve minimum değerlerin sayısı
//--- merkez-dışı ki-kare dağılımına uyan bir örnek oluştur
   MathRandomNoncentralChiSquare(nu_par,si_par,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralChiSquare(x2,nu_par,si_par,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- X ekseninin otomatik olarak ölçeklenmesini engelle
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(NormalizeDouble(max,0));
   graphic.XAxis().Min(min);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
```

```
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityNoncentralChiSquare

Bir rassal x değişkeni için merkez-dışı ki-kare dağılımının olasılık yoğunluk fonksiyonunun değerini nu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityNoncentralChiSquare(
   const double  x,              // rassal değişkenin değeri
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise o
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için merkez-dışı ki-kare dağılımının olasılık yoğunluk fonksiyonunun değerini nu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityNoncentralChiSquare(
   const double  x,              // rassal değişkenin değeri
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için merkez-dışı ki-kare dağılımının olasılık yoğunluk fonksiyonunun değerini nu ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidchisq() fonksiyonunun analoğu

```
bool  MathProbabilityDensityNoncentralChiSquare(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Calculates the value of the probability density function of noncentral chi-squared distribution with the nu parameter for an array of random variables x[]. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityNoncentralChiSquare(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

nu

[in] Dağılım parametresi (serbestlik derecesi).

sigma

[in] Merkezdışılık parametresi.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionNoncentralChiSquare

Bir rassal x değişkeni için merkez-dışı ki-kare dağılımının olasılık fonksiyonunu nu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionNoncentralChiSquare(
   const double  x,              // rassal değişkenin değeri
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   const bool    tail,           // hesplama bayrağı, 'lower_tail=true' ise, x'i aşmaya
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise, d
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için merkez-dışı ki-kare dağılımının olasılık fonksiyonunu nu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionNoncentralChiSquare(
   const double  x,              // rassal değişkenin değeri
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için merkez-dışı ki-kare dağılımının olasılık fonksiyonunu nu ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekipchisq() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionNoncentralChiSquare(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   const bool    tail,           // hesplama bayrağı, 'lower_tail=true' ise, x'i aşmay
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise d
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için merkez-dışı ki-kare dağılımının olasılık fonksiyonunu nu ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionNoncentralChiSquare(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

nu

[in] Dağılım parametresi (serbestlik derecesi).

sigma

[in] Merkezdışılık parametresi.

tail

[in] hesaplama bayrağı. 'true' ise, olasılık rassal değişkenin x'i aşmayan değerleri için hesaplanır.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerinin dizisi.

MathQuantileNoncentralChiSquare

Belirtilen *olasılık* değeri için, ters merkez-dışı ki-kare dağılımının olasılık fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileNoncentralChiSquare(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters merkez-dışı ki-kare dağılımının olasılık fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileNoncentralChiSquare(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters merkez-dışı ki-kare dağılımının olasılık fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qchisq() fonksiyonunun analoğu

```
double  MathQuantileNoncentralChiSquare(
   const double&  probability[],  // rassal değişkenin olasılık değerlerini içeren diz
   const double  nu,              // dağılım parametresi (serbestlik derecesi)
   const double  sigma,           // merkezdışılık parametresi
   const bool    tail,            // hesaplama bayrağı. 'false' ise hesaplama 1.0-olas
   const bool    log_mode,        // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]         // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters merkez-dışı ki-kare dağılımının olasılık fonksiyonunun değerini hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileNoncentralChiSquare(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

nu

[in] Dağılım parametresi (serbestlik derecesi).

sigma

[in] Merkezdışılık parametresi.

tail

[in] Hesaplama bayrağı. 'false' ise hesaplama 1.0 olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomNoncentralChiSquare

nu ve sigma parametrelerini kullanarak, ki-kare dağılımı yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomNoncentralChiSquare(
   const double  nu,            // dağılım parametresi (serbestlik derecesi)
   const double  sigma,         // merkezdışılık parametresi
   int&          error_code     // hata kodu değişkeni
   );
```

nu ve sigma parametrelerini kullanarak, ki-kare dağılımı yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirchisq() fonksiyonunun analoğu

```
bool  MathRandomNoncentralChiSquare(
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   const int     data_count,     // istenen veri miktarı
   double&       result[]        // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

nu

[in] Dağılım parametresi (serbestlik derecesi).

sigma

[in] Merkezdışılık parametresi.

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsNoncentralChiSquare

Merkez-dışı ki-kare dağılımının ilk dört momentinin teorik sayısal değerlerini nu ve sigma parametresi ile hesaplar.

```
double  MathMomentsNoncentralChiSquare(
   const double  nu,              // dağılım parametresi (serbestlik derecesi)
   const double  sigma,           // merkezdışılık parametresi
   double&       mean,            // ortalama değişkeni
   double&       variance,        // varyans değişkeni
   double&       skewness,        // çarpıklık değişkeni
   double&       kurtosis,        // basıklık değişkeni
   int&          error_code       // hata kodu değişkeni
   );
```

Parametreler

nu

[in] Dağılım parametresi (serbestlik derecesi).

sigma

[in] Merkezdışılık parametresi.

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

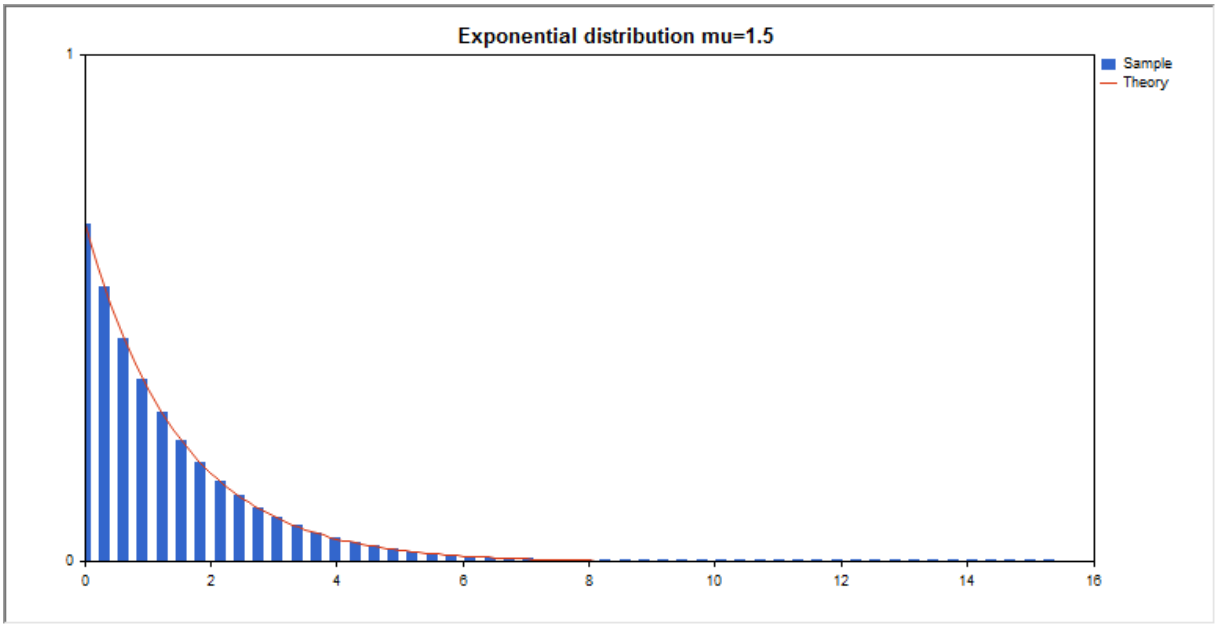
Üssel dağılım

Bu bölüm üssel dağılım ile çalışmak için tasarlanmış fonksiyonlar içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Üssel dağılım şu formülle tanımlanır:

$$f_{Exponential}(x|\mu) = \frac{1}{\mu} e^{-\frac{x}{\mu}}$$

burada:

- x — rassal değişkenin değeri
- μ — beklenen değer



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityExponential	Üssel dağılımın olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionExponential	Üssel dağılımın olasılık fonksiyonunun değerini hesaplar
MathQuantileExponential	Belli bir olasılık değeri için ters üssel dağılımın olasılık fonksiyonunun değerini hesaplar
MathRandomExponential	Üssel dağılım yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur
MathMomentsExponential	üssel dağılımın ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Exponential.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double mu_par=1.5;    // sebestlik derecesi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // örnekteki değerlerin sayısı
   int ncells=51;       // histogramdaki aralık sayısı
   double x[];          // histogram aralıklarının merkezleri
   double y[];          // aralığın içine düşen değerlerin sayısı
   double data[];       // rassal değişken örneği
   double max,min;      // örnekteki maksimum ve minimum değerlerin sayısı
//--- üssel dağılıma uyan bir örnek oluştur
   MathRandomExponential(mu_par,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityExponential(x2,mu_par,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
```

```
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityExponential

Bir rassal x değişkeni için üssel dağılımın olasılık yoğunluk fonksiyonunun değerini mu parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityExponential(
   const double  x,              // rassal değişkenin değeri
   const double  mu,             // dağılım parametresi (beklenen değer)
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise o
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için üssel dağılımın olasılık yoğunluk fonksiyonunun değerini mu parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityExponential(
   const double  x,              // rassal değişkenin değeri
   const double  mu,             // dağılım parametresi (beklenen değer)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için üssel dağılımın olasılık yoğunluk fonksiyonunun değerini mu parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidexp() fonksiyonunun analoğu

```
bool  MathProbabilityDensityExponential(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  mu,             // dağılım parametresi (beklenen değer)
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için üssel dağılımın olasılık yoğunluk fonksiyonunun değerini mu parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityExponential(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  mu,             // dağılım parametresi (beklenen değer)
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

mu

[in] Dağılım parametresi (beklenen değer)

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionExponential

Bir rassal x değişkeni için üssel dağılımın olasılık fonksiyonunu mu parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionExponential(
   const double  x,              // rassal değişkenin değeri
   const double  mu,             // dağılım parametresi (beklenen değer)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal de
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise, d
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için üssel dağılımın olasılık fonksiyonunu mu parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionExponential(
   const double  x,              // rassal değişkenin değeri
   const double  mu,             // dağılım parametresi (beklenen değer)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için mu parametresi ile üssel dağılımın olasılık fonksiyonunu hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekipexp() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionExponential(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  mu,             // dağılım parametresi (beklenen değer)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal d
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise d
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için mu parametresi ile üssel dağılımın olasılık fonksiyonunu hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionExponential(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  mu,             // dağılım parametresi (beklenen değer)
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

mu

[in] Dağılım parametresi (beklenen değer).

tail

[in] hesaplama bayrağı. 'true' ise, olasılık rassal değişkenin x'i aşmayan değerleri için hesaplanır.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerinin dizisi.

MathQuantileExponential

Belirtilen *olasılık* değeri için, ters üssel dağılımın olasılık fonksiyonunun değerini mu parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileExponential(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  mu,             // dağılım parametresi (beklenen değer)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters üssel dağılımın olasılık fonksiyonunun değerini mu parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileExponential(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  mu,             // dağılım parametresi (beklenen değer)
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters üssel dağılımın olasılık fonksiyonunun değerini mu parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qexp() fonksiyonunun analoğu

```
double  MathQuantileExponential(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  mu,             // dağılım parametresi (beklenen değer)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters üssel dağılımın olasılık fonksiyonunun değerini mu parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileExponential(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  mu,             // dağılım parametresi (beklenen değer)
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

mu

[in] Dağılım parametresi (beklenen değer).

tail

[in] Hesaplama bayrağı. 'false' ise hesaplama 1.0 olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomExponential

mu parametresini kullanarak, üssel dağılım yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomExponential(
   const double  mu,            // dağılım parametresi (beklenen değer)
   int&          error_code     // hata kodu değişkeni
   );
```

mu parametresine göre, üssel dağılım yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirexp() fonksiyonunun analoğu

```
bool  MathRandomExponential(
   const double  mu,             // dağılım parametresi (beklenen değer)
   const int     data_count,     // istenen veri miktarı
   double&       result[]        // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

mu

[in] Dağılım parametresi (beklenen değer).

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsExponential

Üssel dağılımın ilk dört momentinin teorik sayısal değerlerini mu parametresine göre hesaplar.

```
double  MathMomentsExponential(
   const double  mu,              // dağılım parametresi (beklenen değer)
   double&       mean,            // ortalama değişkeni
   double&       variance,        // varyans değişkeni
   double&       skewness,        // çarpıklık değişkeni
   double&       kurtosis,        // basıklık değişkeni
   int&          error_code       // hata kodu değişkeni
   );
```

Parametreler

mu

[in] Dağılım parametresi (beklenen değer).

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

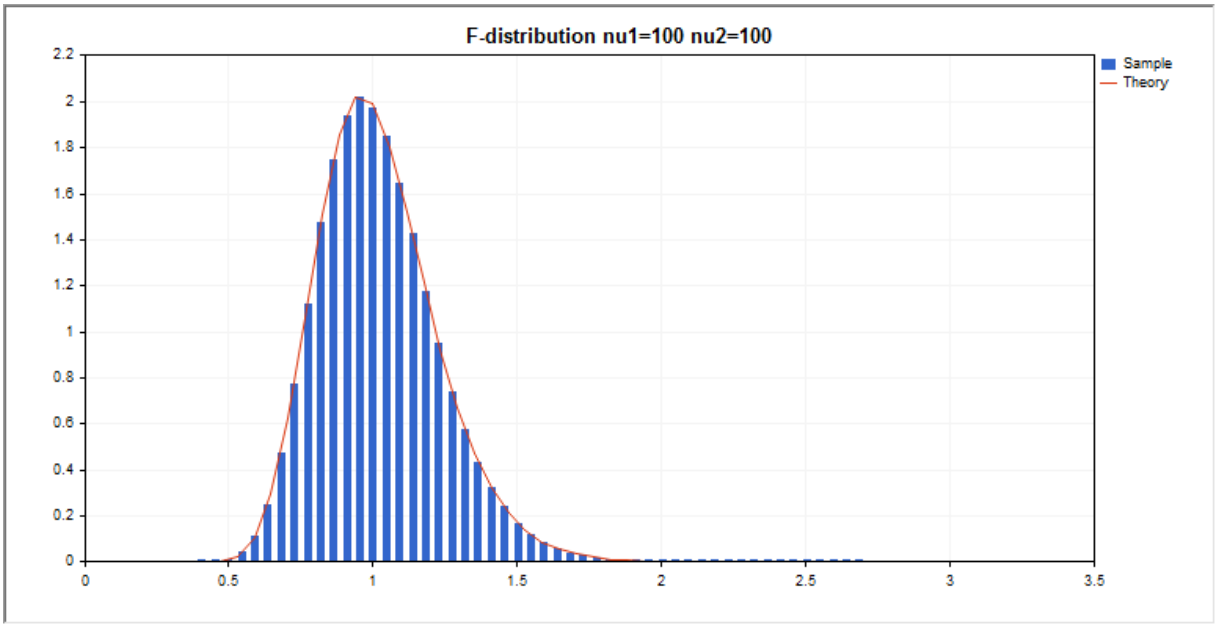
F-dağılımı

Bu bölüm F-dağılımı ile çalışmak için tasarlanmış fonksiyonlar içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. F-dağılımı şu formülle tanımlanır:

$$f_F(x \mid \nu_1, \nu_2) = \frac{\Gamma\left(\frac{\nu_1+\nu_2}{2}\right)}{\Gamma\left(\frac{\nu_1}{2}\right)\Gamma\left(\frac{\nu_2}{2}\right)}\left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_1}{2}} \frac{x^{\frac{\nu_1-2}{2}}}{\left(1+\left(\frac{\nu_1}{\nu_2}\right)x\right)^{\frac{\nu_1+\nu_2}{2}}}$$

burada:

- x — rassal değişkenin değeri
- ν1 — birinci dağılım parametresi (serbestlik derecesi)
- ν2 — ikinci dağılım parametresi (serbestlik derecesi)



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityF	F-dağılımının olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionF	F-dağılımın olasılık fonksiyonunun değerini hesaplar
MathQuantileF	Belli bir olasılık değeri için ters F-dağılımının olasılık fonksiyonunun değerini hesaplar
MathRandomF	F-dağılımı yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur

Fonksiyon	Açıklama
MathMomentsF	Fihser'ın F-dağılımının ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\F.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double nu_1=100;     // ilk serbestlik derecesi
input double nu_2=100;     // ikinci serbestlik derecesi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // örnekteki değerlerin sayısı
   int ncells=51;        // histogramdaki aralık sayısı
   double x[];           // histogram aralıklarının merkezleri
   double y[];           // aralığın içine düşen değerlerin sayısı
   double data[];        // rassal değişken örneği
   double max,min;       // örnekteki maksimum ve minimum değerlerin sayısı
//--- Fisher F-dağılımına uyan bir örnek oluştur
   MathRandomF(nu_1,nu_2,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityF(x2,nu_1,nu_2,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(4);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

```
//+------------------------------------------------------------------+
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityF

Bir rassal x değişkeni için F-dağılımının olasılık yoğunluk fonksiyonunun değerini nu1 ve nu2 parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityF(
   const double  x,              // rassal değişkenin değeri
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise o
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için F-dağılımının olasılık yoğunluk fonksiyonunun değerini nu1 ve nu2 parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityF(
   const double  x,              // rassal değişkenin değeri
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için F-dağılımının olasılık fonksiyonunun değerini nu1 ve nu2 parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidf() fonksiyonunun analoğu

```
bool  MathProbabilityDensityF(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log_
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için F-dağılımının olasılık fonksiyonunun değerini nu1 ve nu2 parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityF(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

nu1

[in] Dağılımın ilk parametresi (serbestlik derecesi).

nu2

[in] Dağılımın ikinci parametresi (serbestlik derecesi).

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionF

Bir rassal x değişkeni için F-dağılımının olasılık fonksiyonunun değerini nu1 ve nu2 parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionF(
   const double  x,              // rassal değişkenin değeri
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal de
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise, d
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için F-dağılımının olasılık fonksiyonunun değerini nu1 ve nu2 parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionF(
   const double  x,              // rassal değişkenin değeri
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için F-dağılımının olasılık fonksiyonunu nu1 ve nu2 parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekipf() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionF(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal d
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise,
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için F-dağılımının olasılık fonksiyonunu nu1 ve nu2 parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionF(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

nu1

[in] Dağılımın ilk parametresi (serbestlik derecesi).

nu2

[in] Dağılımın ikinci parametresi (serbestlik derecesi).

tail

[in] hesaplama bayrağı. 'true' ise, olasılık rassal değişkenin x'i aşmayan değerleri için hesaplanır.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerinin dizisi.

MathQuantileF

Belirtilen *olasılık* değeri için, ters F-dağılımının olasılık fonksiyonunun değerini nu1 ve nu2 parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileF(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters F-dağılımının olasılık fonksiyonunun değerini nu1 ve nu2 parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileF(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters F-dağılımının olasılık fonksiyonunun değerini nu1 ve nu2 parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qf() fonksiyonunun analoğu

```
double  MathQuantileF(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters F-dağılımının olasılık fonksiyonunun değerini nu1 ve nu2 parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileF(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

nu1

[in] Dağılımın ilk parametresi (serbestlik derecesi).

nu2

[in] Dağılımın ikinci parametresi (serbestlik derecesi).

tail

[in] Hesaplama bayrağı. lower_tail=false ise hesaplama 1.0-olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomF

nu1 ve nu2 parametrelerine göre, F-dağılımı yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomF(
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   int&          error_code      // hata kodu değişkeni
   );
```

nu1 ve nu2 parametrelerine göre, F-dağılımı yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirf() fonksiyonunun analoğu

```
bool  MathRandomF(
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const int     data_count,     // istenen veri miktarı
   double&       result[]        // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

nu1

[in] Dağılımın ilk parametresi (serbestlik derecesi).

nu2

[in] Dağılımın ikinci parametresi (serbestlik derecesi).

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsF

Fihser'ın F-dağılımının ilk dört momentinin teorik sayısal değerlerini nu1 ve nu2 parametrelerine göre hesaplar.

```
double  MathMomentsF(
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   double&       mean,           // ortalama değişkeni
   double&       variance,       // varyans değişkeni
   double&       skewness,       // çarpıklık değişkeni
   double&       kurtosis,       // basıklık değişkeni
   int&          error_code      // hata kodu değişkeni
   );
```

Parametreler

nu1

[in] Dağılımın ilk parametresi (serbestlik derecesi).

nu2

[in] Dağılımın ikinci parametresi (serbestlik derecesi).

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

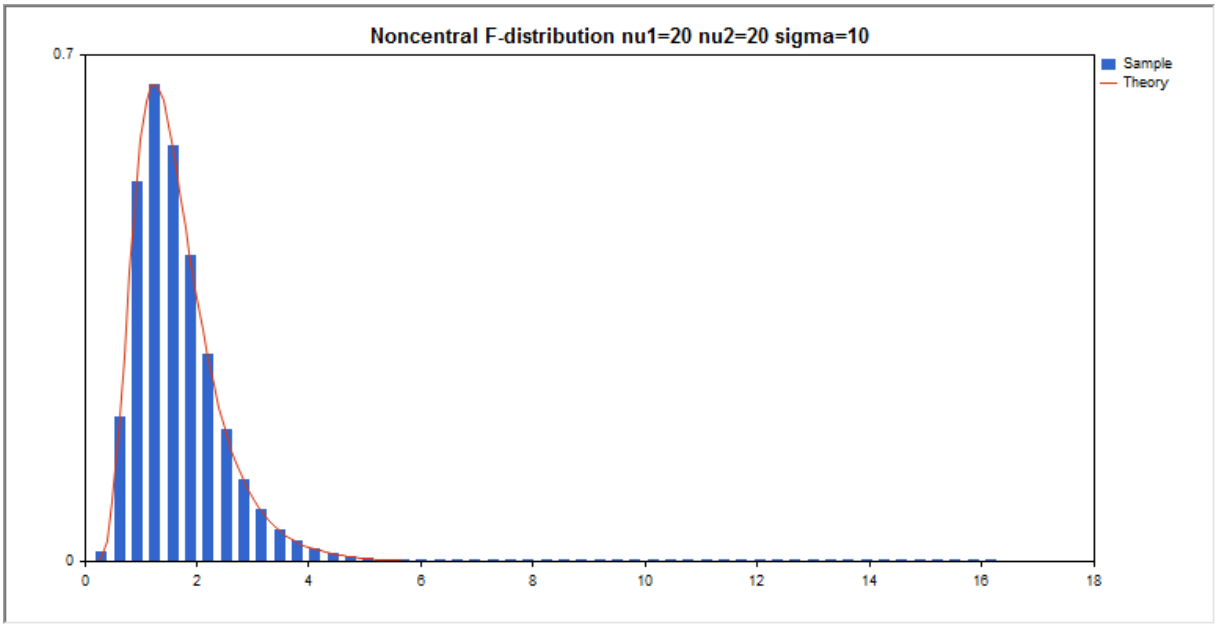
Merkez-dışı F-dağılımı

Bu bölüm merkez-dışı F-dağılımı ile çalışmak için tasarlanmış fonksiyonlar içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Merkez-dışı F-dağılımı şu formülle tanımlanır:

$$f_{NoncentralF}(x|\nu_1,\nu_2,\sigma) = e^{-\frac{\sigma}{2}} \sum_{r=0}^{\infty} \frac{1}{r!}\left(\frac{\sigma}{2}\right)^r \frac{\Gamma\left(\frac{\nu_1+\nu_2}{2}+r\right)}{\Gamma\left(\frac{\nu_2}{2}+r\right)\Gamma\left(\frac{\nu_2}{2}\right)}\left(\frac{\nu_1}{\nu_2}\right)^{\frac{\nu_2}{2}+r} \frac{x^{\frac{\nu_2}{2}-1+r}}{\left(1+\frac{\nu_1}{\nu_2}x\right)^{\frac{\nu_1+\nu_2}{2}+r}}$$

burada:

- x — rassal değişkenin değeri
- ν1 — birinci dağılım parametresi (serbestlik derecesi)
- ν2 — ikinci dağılım parametresi (serbestlik derecesi)
- σ — merkez-dışılık parametresi



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityNoncentralF	Merkez-dışı F-dağılımının olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionNoncentralF	Merkez-dışı F-dağılımın olasılık fonksiyonunun değerini hesaplar
MathQuantileNoncentralF	Belli bir olasılık değeri için ters merkez-dışı F-dağılımının olasılık fonksiyonunun değerini hesaplar

Fonksiyon	Açıklama
MathRandomNoncentralF	Merkez-dışı F-dağılımı yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur
MathMomentsNoncentralF	Merkez-dışı F-dağılımının ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralF.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double nu_1=20;     // ilk serbestlik derecesi
input double nu_2=20;     // ikinci serbestlik derecesi
input double sig=10;      // merkez-dışılık parametresi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;         // örnekteki değerlerin sayısı
   int ncells=51;         // histogramdaki aralık sayısı
   double x[];            // histogram aralıklarının merkezleri
   double y[];            // aralığın içine düşen değerlerin sayısı
   double data[];         // rassal değişken örneği
   double max,min;        // örnekteki maksimum ve minimum değerlerin sayısı
//--- merkez-dışı Fisher F-dağılımına uyan bir örnek oluştur
   MathRandomNoncentralF(nu_1,nu_2,sig,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralF(x2,nu_1,nu_2,sig,false,y2);
```

```
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
```

```
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityNoncentralF

Bir rassal x değişkeni için merkez-dışı F-dağılımının olasılık yoğunluk fonksiyonunun değerini nu1, nu2 ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityNoncentralF(
   const double  x,              // rassal değişkenin değeri
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise o
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için merkez-dışı F-dağılımının olasılık yoğunluk fonksiyonunun değerini nu1, nu2 ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityNoncentralF(
   const double  x,              // rassal değişkenin değeri
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için merkez-dışı F-dağılımının olasılık yoğunluk fonksiyonunun değerini nu1, nu2 ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidf() fonksiyonunun analoğu

```
bool  MathProbabilityDensityNoncentralF(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log_
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için merkez-dışı F-dağılımının olasılık yoğunluk fonksiyonunun değerini nu1, nu2 ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityNoncentralF(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

nu1

[in] Dağılımın ilk parametresi (serbestlik derecesi).

nu2

[in] Dağılımın ikinci parametresi (serbestlik derecesi).

sigma

[in] Merkezdışılık parametresi.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionNoncentralF

Bir rassal x değişkeni için merkez-dışı F-dağılımının olasılık fonksiyonunu nu1, nu2 ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionNoncentralF(
   const double  x,              // rassal değişkenin değeri
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal de
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise ol
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için merkez-dışı F-dağılımının olasılık fonksiyonunu nu1, nu2 ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionNoncentralF(
   const double  x,              // rassal değişkenin değeri
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için merkez-dışı F-dağılımının olasılık fonksiyonunu nu1, nu2 ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekipf() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionNoncentralF(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal d
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise o
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için merkez-dışı F-dağılımının olasılık fonksiyonunu nu1, nu2 ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionNoncentralF(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

nu1

[in] Dağılımın ilk parametresi (serbestlik derecesi).

nu2

[in] Dağılımın ikinci parametresi (serbestlik derecesi).

sigma

[in] Merkezdışılık parametresi.

tail

[in] hesaplama bayrağı. 'true' ise, olasılık rassal değişkenin x'i aşmayan değerleri için hesaplanır.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerinin dizisi.

MathQuantileF

Belirtilen *olasılık* değeri için, merkez-dışı ters F-dağılımının olasılık fonksiyonunun değerini nu1, nu2 ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileF(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, merkez-dışı ters F-dağılımının olasılık fonksiyonunun değerini nu1, nu2 ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileF(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters merkez-dışı F-dağılımının olasılık fonksiyonunun değerini nu1, nu2 ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qf() fonksiyonunun analoğu

```
double  MathQuantileF(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters merkez-dışı F-dağılımının olasılık fonksiyonunun değerini nu1, nu2 ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileF(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

nu1

[in] Dağılımın ilk parametresi (serbestlik derecesi).

nu2

[in] Dağılımın ikinci parametresi (serbestlik derecesi).

sigma

[in] Merkezdışılık parametresi.

tail

[in] Hesaplama bayrağı. 'false' ise hesaplama 1.0 olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomNoncentralF

nu1, nu2 ve sigma parametrelerine göre, merkez-dışı F-dağılımı yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomNoncentralF(
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

nu1, nu2 ve sigma parametrelerine göre, merkez-dışı F-dağılımı yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirf() fonksiyonunun analoğu

```
bool  MathRandomNoncentralF(
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   const int     data_count,     // istenen veri miktarı
   double&       result[]        // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

nu1

[in] Dağılımın ilk parametresi (serbestlik derecesi).

nu2

[in] Dağılımın ikinci parametresi (serbestlik derecesi).

sigma

[in] Merkezdışılık parametresi.

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsNoncentralF

Merkez-dışı F-dağılımının ilk dört momentinin teorik sayısal değerlerini nu1, nu2 ve sigma parametrelerine göre hesaplar.

```
double  MathMomentsNoncentralF(
   const double  nu1,            // dağılımın ilk parametresi (serbestlik derecesi)
   const double  nu2,            // dağılımın ikinci parametresi (serbestlik derecesi)
   const double  sigma,          // merkezdışılık parametresi
   double&       mean,           // ortalama değişkeni
   double&       variance,       // varyans değişkeni
   double&       skewness,       // çarpıklık değişkeni
   double&       kurtosis,       // basıklık değişkeni
   int&          error_code      // hata kodu değişkeni
   );
```

Parametreler

nu1

[in] Dağılımın ilk parametresi (serbestlik derecesi).

nu2

[in] Dağılımın ikinci parametresi (serbestlik derecesi).

sigma

[in] Merkezdışılık parametresi.

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

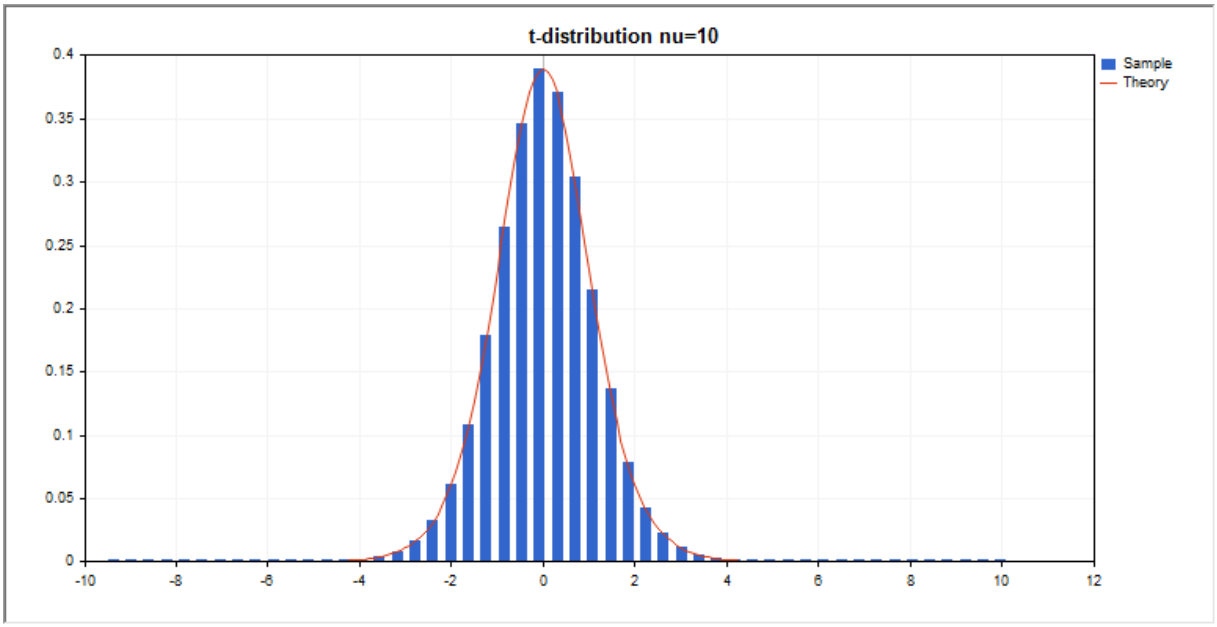
t-dağılımı

Bu bölüm Student t-dağılımı ile çalışmak için gerekli olan fonksiyonları içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. t-dağılımı şu formülle tanımlanır:

$$f_T(x|\nu) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \frac{1}{\sqrt{\pi\nu}} \frac{1}{\left(1+\frac{x^2}{\nu}\right)^{\frac{\nu+1}{2}}}$$

burada:

- x — rassal değişkenin değeri
- ν — dağılım parametresi (serbestlik derecesi)



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityT	t-dağılımının olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionT	t-dağılımıın olasılık fonksiyonunun değerini hesaplar
MathQuantileT	Belli bir olasılık değeri için ters t-dağılımının olasılık fonksiyonunun değerini hesaplar
MathRandomT	t-dağılımı yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur
MathMomentsT	t-dağılımının ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\T.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double nu_par=10;    // sebestlik derecesi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // örnekteki değerlerin sayısı
   int ncells=51;       // histogramdaki aralık sayısı
   double x[];          // histogram aralıklarının merkezleri
   double y[];          // aralığın içine düşen değerlerin sayısı
   double data[];       // rassal değişken örneği
   double max,min;      // örnekteki maksimum ve minimum değerlerin sayısı
//--- t dağılımına uyan bir örnek oluştur
   MathRandomT(nu_par,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityT(x2,nu_par,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
```

```
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityT

Bir rassal x değişkeni için t-dağılımının olasılık yoğunluk fonksiyonunun değerini nu parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityT(
   const double  x,              // rassal değişkenin değeri
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise o
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için t-dağılımının olasılık yoğunluk fonksiyonunun değerini nu parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityT(
   const double  x,              // rassal değişkenin değeri
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için t-dağılımının olasılık yoğunluk fonksiyonunun değerini nu parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidt() fonksiyonunun analoğu

```
bool  MathProbabilityDensityT(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için t-dağılımının olasılık yoğunluk fonksiyonunun değerini nu parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityT(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

nu

[in] Dağılım parametresi (serbestlik derecesi).

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionT

Bir rassal x değişkeni için t-dağılımının olasılık fonksiyonunun değerini nu parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionT(
   const double  x,              // rassal değişkenin değeri
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal de
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise ol
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için t-dağılımının olasılık fonksiyonunun değerini nu parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionT(
   const double  x,              // rassal değişkenin değeri
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için t-dağılımının olasılık fonksiyonunun değerini nu parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekipt() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionT(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal d
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise d
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için t-dağılımının olasılık fonksiyonunun değerini nu parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionT(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

nu

[in] Dağılım parametresi (serbestlik derecesi).

tail

[in] hesaplama bayrağı. 'true' ise, olasılık rassal değişkenin x'i aşmayan değerleri için hesaplanır.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerinin dizisi.

MathQuantileT

Belirtilen *olasılık* değeri için, ters t-dağılımının olasılık fonksiyonunun değerini nu parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileT(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters t-dağılımının olasılık fonksiyonunun değerini nu parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileT(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters t- dağılımının olasılık fonksiyonunun değerini mu parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qt() fonksiyonunun analoğu

```
double  MathQuantileT(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters t- dağılımının olasılık fonksiyonunun değerini mu parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileT(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

nu

[in] Dağılım parametresi (serbestlik derecesi).

tail

[in] Hesaplama bayrağı. 'false' ise hesaplama 1.0 olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomT

nu parametresini kullanarak, t-dağılımı yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomT(
   const double  nu,              // dağılım parametresi (serbestlik derecesi)
   int&          error_code       // hata kodu değişkeni
   );
```

nu parametresini kullanarak, t-dağılımı yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirt() fonksiyonunun analoğu

```
bool  MathRandomT(
   const double  nu,              // dağılım parametresi (serbestlik derecesi)
   const int     data_count,      // istenen veri miktarı
   double&       result[]         // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

nu

[in] Dağılım parametresi (serbestlik derecesi).

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsT

Student t-dağılımının ilk dört momentinin teorik sayısal değerlerini nu parametresi ile hesaplar.

```
double  MathMomentsT(
   const double  nu,              // dağılım parametresi (serbestlik derecesi)
   double&       mean,            // ortalama değişkeni
   double&       variance,        // varyans değişkeni
   double&       skewness,        // çarpıklık değişkeni
   double&       kurtosis,        // basıklık değişkeni
   int&          error_code       // hata kodu değişkeni
   );
```

Parametreler

nu

[in] Dağılım parametresi (serbestlik derecesi).

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

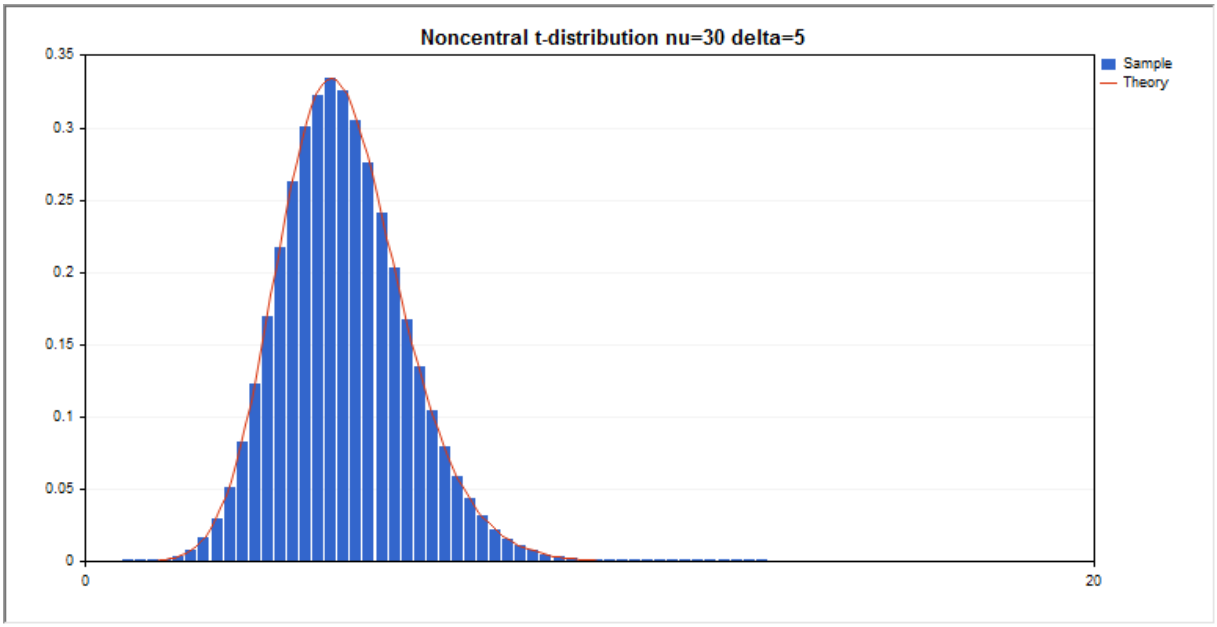
t-dağılımı

Bu bölüm merkez-dışı Student t-dağılımı ile çalışmak için gerekli olan fonksiyonları içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Merkez-dışı t-dağılımı şu formülle tanımlanır:

$$f_{NoncentralT}(x|\nu,\delta)=\frac{\nu^{\frac{\nu}{2}}e^{-\frac{\sigma^2}{2}}}{\Gamma\left(\frac{\nu}{2}\right)\sqrt{\pi}\left(\nu+x^2\right)^{\frac{\nu+1}{2}}}\sum_{r=0}^{\infty}\frac{(x\delta)^r}{r!}\left(\frac{2}{\nu+x^2}\right)^{\frac{r}{2}}\Gamma\left(\frac{\nu+r+1}{2}\right)$$

burada:

- x — rassal değişkenin değeri
- ν — dağılım parametresi (serbestlik derecesi)
- σ — merkez-dışılık parametresi



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityNoncentralT	Merkez-dışı t-dağılımının olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionNoncentralT	Merkez-dışı t-dağılımın olasılık fonksiyonunun değerini hesaplar
MathQuantileNoncentralT	Belli bir olasılık değeri için ters merkez-dışı t-dağılımının olasılık fonksiyonunun değerini hesaplar
MathRandomNoncentralT	Merkez-dışı t-dağılımı yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur

Fonksiyon	Açıklama
MathMomentsNoncentralT	Merkez-dışı t-dağılımının ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NoncentralT.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double nu_par=30;        // sebestlik derecesi
input double delta_par=5;     // merkez-dışılık parametresi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // örnekteki değerlerin sayısı
   int ncells=51;        // histogramdaki aralık sayısı
   double x[];           // histogram aralıklarının merkezleri
   double y[];           // aralığın içine düşen değerlerin sayısı
   double data[];        // rassal değişken örneği
   double max,min;       // örnekteki maksimum ve minimum değerlerin sayısı
//--- merkezdışı t dağılımına uyan bir örnek oluştur
   MathRandomNoncentralT(nu_par,delta_par,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityNoncentralT(x2,nu_par,delta_par,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
```

```
//+------------------------------------------------------------------+
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityNoncentralT

Bir rassal x değişkeni için merkez-dışı t-dağılımının olasılık yoğunluk fonksiyonunun değerini nu ve delta parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityNoncentralT(
   const double  x,              // rassal değişkenin değeri
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  delta,          // merkezdışılık parametresi
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise o
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için merkez-dışı t-dağılımının olasılık yoğunluk fonksiyonunun değerini nu ve delta parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityNoncentralT(
   const double  x,              // rassal değişkenin değeri
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  delta,          // merkezdışılık parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için merkez-dışı t-dağılımının olasılık yoğunluk fonksiyonunun değerini nu ve delta parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidt() fonksiyonunun analoğu

```
bool  MathProbabilityDensityNoncentralT(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  delta,          // merkezdışılık parametresi
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log_
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için merkez-dışı t-dağılımının olasılık yoğunluk fonksiyonunun değerini nu ve delta parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityNoncentralT(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  delta,          // merkezdışılık parametresi
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

nu

[in] Dağılım parametresi (serbestlik derecesi).

delta

[in] Merkezdışılık parametresi.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionNoncentralT

Bir rassal x değişkeni için merkez-dışı t-dağılımının olasılık fonksiyonunu nu ve delta parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionNoncentralT(
   const double  x,              // rassal değişkenin değeri
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  delta,          // merkezdışılık parametresi
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal de
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise ol
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için merkez-dışı t-dağılımının olasılık fonksiyonunu nu ve delta parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionNoncentralT(
   const double  x,              // rassal değişkenin değeri
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  delta,          // merkezdışılık parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için merkez-dışı t-dağılımının olasılık fonksiyonunu nu ve delta parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekipt() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionNoncentralT(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  delta,          // merkezdışılık parametresi
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal d
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise d
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için merkez-dışı t-dağılımının olasılık fonksiyonunu nu ve delta parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionNoncentralT(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  delta,          // merkezdışılık parametresi
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

nu

[in] Dağılım parametresi (serbestlik derecesi).

delta

[in] Merkezdışılık parametresi.

tail

[in] hesaplama bayrağı. 'true' ise, olasılık rassal değişkenin x'i aşmayan değerleri için hesaplanır.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerinin dizisi.

MathQuantileNoncentralT

Belirtilen *olasılık* değeri için, ters merkez-dışı t-dağılımının olasılık fonksiyonunun değerini nu ve delta parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileNoncentralT(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  delta,          // merkezdışılık parametresi
   const bool    tail,           // hesaplama bayrağı. lower_tail=false ise hesaplama
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters merkez-dışı t-dağılımının olasılık fonksiyonunun değerini nu ve delta parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileNoncentralT(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  delta,          // merkezdışılık parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters merkez-dışı t- dağılımının olasılık fonksiyonunun değerini mu ve delta parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qt() fonksiyonunun analoğu

```
double  MathQuantileNoncentralT(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren diz
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  delta,          // merkezdışılık parametresi
   const bool    tail,           // hesaplama bayrağı. lower_tail=false ise hesaplama
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters merkez-dışı t- dağılımının olasılık fonksiyonunun değerini mu ve delta parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileNoncentralT(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren diz
   const double  nu,             // dağılım parametresi (serbestlik derecesi)
   const double  delta,          // merkezdışılık parametresi
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

nu

[in] Dağılım parametresi (serbestlik derecesi).

delta

[in] Merkezdışılık parametresi.

tail

[in] Hesaplama bayrağı. 'false' ise hesaplama 1.0 olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomNoncentralT

nu ve delta parametrelerini kullanarak, merkez-dışı t-dağılımı yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomNoncentralT(
   const double  nu,              // dağılım parametresi (serbestlik derecesi)
   const double  delta,           // merkezdışılık parametresi
   int&          error_code       // hata kodu değişkeni
   );
```

nu ve delta parametrelerini kullanarak, merkez-dışı t-dağılımı yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirt() fonksiyonunun analoğu

```
bool  MathRandomNoncentralT(
   const double  nu,              // dağılım parametresi (serbestlik derecesi)
   const double  delta,           // merkezdışılık parametresi
   const int     data_count,      // istenen veri miktarı
   double&       result[]         // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

nu

[in] Dağılım parametresi (serbestlik derecesi).

delta

[in] Merkezdışılık parametresi.

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsNoncentralT

Merkez-dışı t-dağılımının ilk dört momentinin teorik sayısal değerlerini nu ve delta parametresi ile hesaplar.

```
double  MathMomentsNoncentralT(
   const double  nu,              // dağılım parametresi (serbestlik derecesi)
   const double  delta,           // merkezdışılık parametresi
   double&       mean,            // ortalama değişkeni
   double&       variance,        // varyans değişkeni
   double&       skewness,        // çarpıklık değişkeni
   double&       kurtosis,        // basıklık değişkeni
   int&          error_code       // hata kodu değişkeni
   );
```

Parametreler

nu

[in] Dağılım parametresi (serbestlik derecesi).

delta

[in] Merkezdışılık parametresi.

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

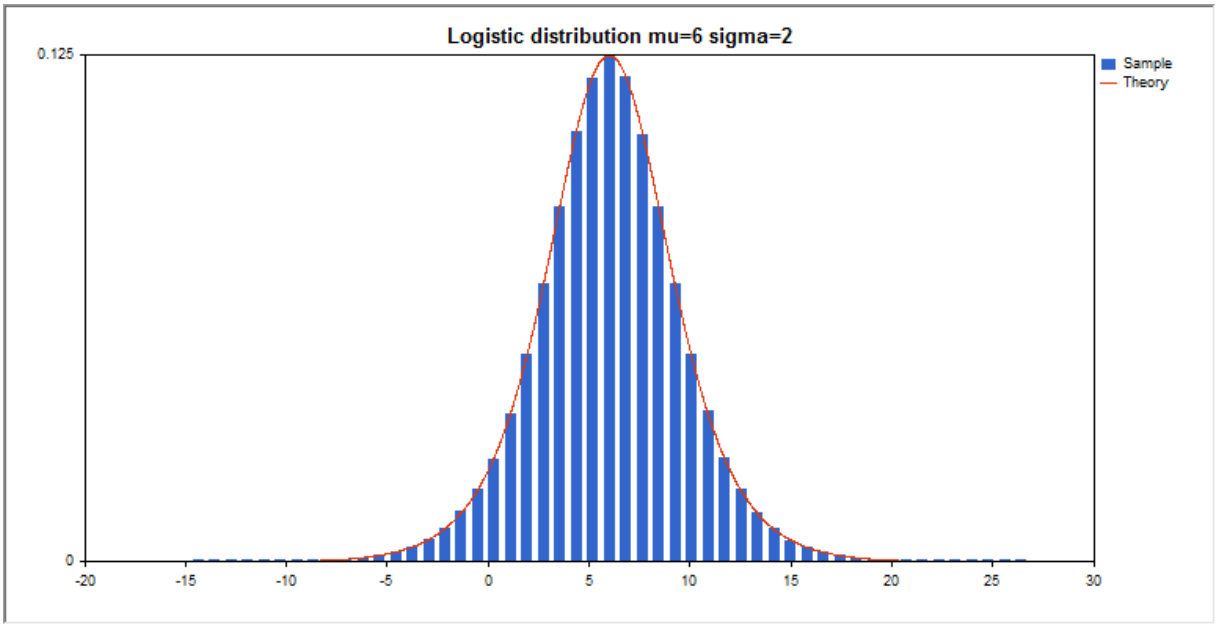
Lojistik dağılım

Bu bölüm lojistik dağılım ile çalışmak için tasarlanmış fonksiyonlar içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Lojistik dağılım şu formülle tanımlanır:

$$f_{Logistic}(x|\mu,\sigma) = \frac{e^{\frac{x-\mu}{\sigma}}}{\sigma\left(1+e^{\frac{x-\mu}{\sigma}}\right)^2}$$

burada:

- x — rassal değişkenin değeri
- μ — dağılımın ortalama parametresi
- σ — dağılımın ölçek parametresi



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityLogistic	Lojistik dağılımın olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionLogistic	Lojistik dağılımın olasılık fonksiyonunun değerini hesaplar
MathQuantileLogistic	Belli bir olasılık değeri için ters lojistik dağılımın olasılık fonksiyonunun değerini hesaplar
MathRandomLogistic	Lojistik dağılım yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur

Fonksiyon	Açıklama
MathMomentsLogistic	Lojistik dağılımın ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Logistic.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double mu_par=6;         // dağılımın ortalama parametresi
input double sigma_par=2;      // dağılımın ölçek parametresi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;        // örnekteki değerlerin sayısı
   int ncells=51;        // histogramdaki aralık sayısı
   double x[];           // histogram aralıklarının merkezleri
   double y[];           // aralığın içine düşen değerlerin sayısı
   double data[];        // rassal değişken örneği
   double max,min;       // örnekteki maksimum ve minimum değerlerin sayısı
//--- lojistik dağılıma uyan bir örnek oluştur
   MathRandomLogistic(mu_par,sigma_par,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityLogistic(x2,mu_par,sigma_par,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- Y ekseninin otomatik olarak ölçeklenmesini engelle
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(theor_max);
   graphic.YAxis().Min(0);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
```

```
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityLogistic

Bir rassal x değişkeni için lojistik dağılımın olasılık yoğunluk fonksiyonunun değerini mu ve sigma parametreline göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityLogistic(
   const double  x,              // rassal değişkenin değeri
   const double  mu,             //dağılımın ortalama parametresi
   const double  sigma,          // dağılımın ölçek parametresi
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise o
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için lojistik dağılımın olasılık yoğunluk fonksiyonunun değerini mu ve sigma parametreline göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityLogistic(
   const double  x,              // rassal değişkenin değeri
   const double  mu,             //dağılımın ortalama parametresi
   const double  sigma,          // dağılımın ölçek parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için lojistik dağılımın olasılık yoğunluk fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidlogis() fonksiyonunun analoğu

```
bool  MathProbabilityDensityLogistic(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  mu,             //dağılımın ortalama parametresi
   const double  sigma,          // dağılımın ölçek parametresi
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log_
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için lojistik dağılımın olasılık yoğunluk fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityLogistic(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  mu,             //dağılımın ortalama parametresi
   const double  sigma,          // dağılımın ölçek parametresi
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

mu

[in] dağılımın ortalama parametresi.

sigma

[in] dağılımın ölçek parametresi.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionLogistic

Bir rassal x değişkeni için lojistik dağılımın olasılık fonksiyonunu mu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionLogistic(
   const double  x,              // rassal değişkenin değeri
   const double  mu,             //dağılımın ortalama parametresi
   const double  sigma,          // dağılımın ölçek parametresi
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal de
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise ol
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için lojistik dağılımın olasılık fonksiyonunu mu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionLogistic(
   const double  x,              // rassal değişkenin değeri
   const double  mu,             //dağılımın ortalama parametresi
   const double  sigma,          // dağılımın ölçek parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için mu ve sigma parametreleri ile lojistik dağılımın olasılık fonksiyonunu hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionLogistic(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  mu,             //dağılımın ortalama parametresi
   const double  sigma,          // dağılımın ölçek parametresi
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal d
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise d
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için mu ve sigma parametreleri ile lojistik dağılımın olasılık fonksiyonunu hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekiplogis() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionLogistic(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  mu,             //dağılımın ortalama parametresi
   const double  sigma,          // dağılımın ölçek parametresi
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

mu

[in] dağılımın ortalama parametresi.

sigma

[in] dağılımın ölçek parametresi.

tail

[in] hesaplama bayrağı. 'true' ise, olasılık rassal değişkenin x'i aşmayan değerleri için hesaplanır.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerinin dizisi.

MathQuantileLogistic

Belirtilen *olasılık* değeri için, ters lojistik dağılımın olasılık fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileLogistic(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  mu,             //dağılımın ortalama parametresi
   const double  sigma,          // dağılımın ölçek parametresi
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters lojistik dağılımın olasılık fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileLogistic(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  mu,             //dağılımın ortalama parametresi
   const double  sigma,          // dağılımın ölçek parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters lojistik dağılımın olasılık fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qlogis() fonksiyonunun analoğu

```
double  MathQuantileLogistic(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  mu,             //dağılımın ortalama parametresi
   const double  sigma,          // dağılımın ölçek parametresi
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters lojistik dağılımın olasılık fonksiyonunun değerini mu ve sigma parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileLogistic(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  mu,             //dağılımın ortalama parametresi
   const double  sigma,          // dağılımın ölçek parametresi
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

mu

[in] dağılımın ortalama parametresi.

sigma

[in] dağılımın ölçek parametresi.

tail

[in] Hesaplama bayrağı. 'false' ise hesaplama 1.0 olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomLogistic

mu ve sigma parametrelerini kullanarak, lojistik dağılım yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomLogistic(
   const double  mu,              //dağılımın ortalama parametresi
   const double  sigma,           // dağılımın ölçek parametresi
   int&          error_code       // hata kodu değişkeni
   );
```

mu ve sigma parametrelerini kullanarak, lojistik dağılım yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirlogis() fonksiyonunun analoğu

```
bool  MathRandomLogistic(
   const double  mu,              //dağılımın ortalama parametresi
   const double  sigma,           // dağılımın ölçek parametresi
   const int     data_count,      // istenen veri miktarı
   double&       result[]         // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

mu

[in] dağılımın ortalama parametresi.

sigma

[in] dağılımın ölçek parametresi.

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsLogistic

Lojistik dağılımın ilk dört momentinin teorik sayısal değerlerini mu ve sigma parametrelerine göre hesaplar.

```
double  MathMomentsLogistic(
   const double  mu,              //dağılımın ortalama parametresi
   const double  sigma,           // dağılımın ölçek parametresi
   double&       mean,            // ortalama değişkeni
   double&       variance,        // varyans değişkeni
   double&       skewness,        // çarpıklık değişkeni
   double&       kurtosis,        // basıklık değişkeni
   int&          error_code       // hata kodu değişkeni
   );
```

Parametreler

mu

[in] dağılımın ortalama parametresi.

sigma

[in] dağılımın ölçek parametresi.

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

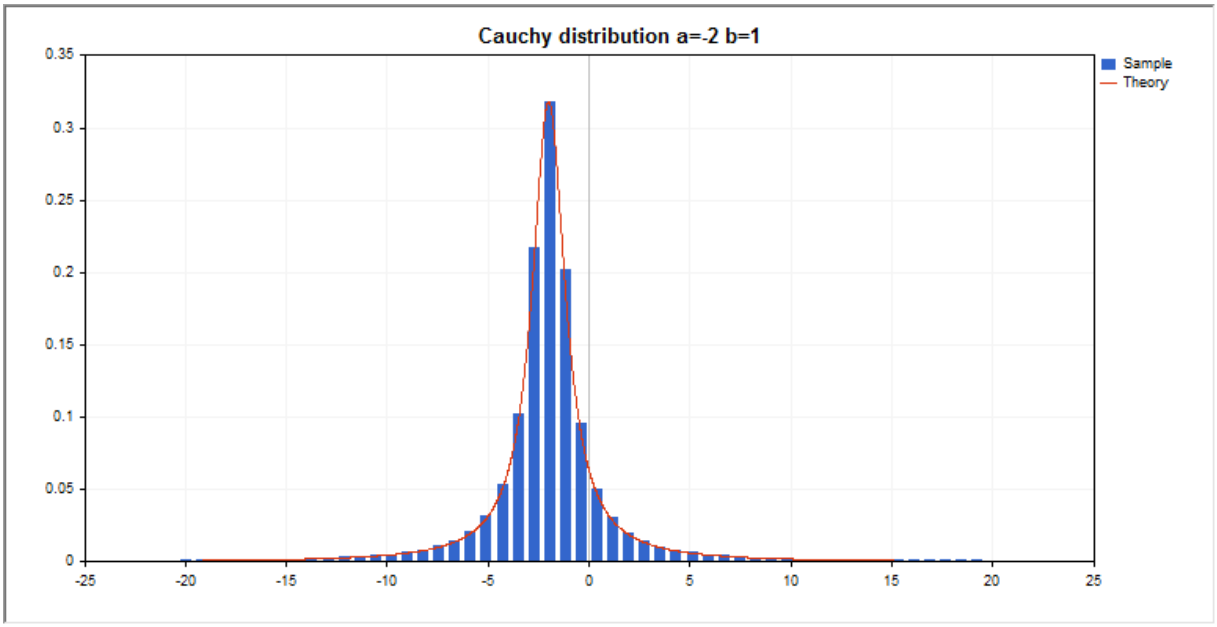
Cauchy dağılımı

Bu bölüm Cauchy dağılımı ile çalışmak için tasarlanmış fonksiyonlar içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Cauchy dağılımı şu formülle tanımlanır:

$$f_{Cauchy}(x|a,b) = \frac{1}{\pi}\frac{b}{\left(b^2+(x-a)^2\right)}$$

burada:

- x — rassal değişkenin değeri
- a — dağılımın ortalama parametresi
- b — dağılımın ölçek parametresi



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityCauchy	Cauchy dağılımının olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionCauchy	Cauchy dağılımının olasılık fonksiyonunun değerini hesaplar
MathQuantileCauchy	Belli bir olasılık değeri için ters Cauchy dağılımının olasılık fonksiyonunun değerini hesaplar
MathRandomCauchy	Cauchy dağılımı yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur
MathMomentsCauchy	Cauchy dağılımının ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Cauchy.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double a_par=-2;      // dağılımın ortalama parametresi
input double b_par=1;       // dağılımın ölçek parametresi
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // örnekteki değerlerin sayısı
   int ncells=51;          // histogramdaki aralık sayısı
   double x[];             // histogram aralıklarının merkezleri
   double y[];             // aralığın içine düşen değerlerin sayısı
   double data[];          // rassal değişken örneği
   double max,min;         // örnekteki maksimum ve minimum değerlerin sayısı
//--- Cauchy dağılımına uygun bir örnek oluştur
   MathRandomCauchy(a_par,b_par,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityCauchy(x2,a_par,b_par,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=(int)MathRound((data[i]-minv)/width);
      if(ind>=0 && ind<cells)
         frequency[ind]++;
     }
   return(true);
```

```
  }
//+------------------------------------------------------------------+
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityCauchy

Bir rassal x değişkeni için, Cauchy dağılımının olasılık yoğunluk fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityCauchy(
   const double  x,              // rassal değişkenin değeri
   const double  a,              //dağılımın ortalama parametresi
   const double  b,              // dağılımın ölçek parametresi
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise o
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için, Cauchy dağılımının olasılık yoğunluk fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityCauchy(
   const double  x,              // rassal değişkenin değeri
   const double  a,              //dağılımın ortalama parametresi
   const double  b,              // dağılımın ölçek parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için Cauchy dağılımının olasılık yoğunluk fonksiyonununun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidcauchy() fonksiyonunun analoğu

```
bool  MathProbabilityDensityCauchy(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // dağılımın ortalama parametresi
   const double  b,              // dağılımın ölçek parametresi
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için Cauchy dağılımının olasılık yoğunluk fonksiyonununun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityCauchy(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // dağılımın ortalama parametresi
   const double  b,              // dağılımın ölçek parametresi
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

a

[in] dağılımın ortalama parametresi.

b

[in] dağılımın ölçek parametresi.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionCauchy

Bir rassal x değişkeni için, Cauchy dağılımının olasılık fonksiyonunu a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionCauchy(
   const double  x,              // rassal değişkenin değeri
   const double  a,              //dağılımın ortalama parametresi
   const double  b,              // dağılımın ölçek parametresi
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal de
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise ol
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için, Cauchy dağılımının olasılık fonksiyonunu a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionCauchy(
   const double  x,              // rassal değişkenin değeri
   const double  a,              //dağılımın ortalama parametresi
   const double  b,              // dağılımın ölçek parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, Cauchy dağılımının olasılık fonksiyonunu a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionCauchy(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // dağılımın ortalama parametresi
   const double  b,              // dağılımın ölçek parametresi
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal d
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise d
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, Cauchy dağılımının olasılık fonksiyonunu a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekiplogis() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionCauchy(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // dağılımın ortalama parametresi
   const double  b,              // dağılımın ölçek parametresi
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

a

[in] dağılımın ortalama parametresi.

b

[in] dağılımın ölçek parametresi.

tail

[in] hesaplama bayrağı. 'true' ise, olasılık rassal değişkenin x'i aşmayan değerleri için hesaplanır.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerinin dizisi.

MathQuantileCauchy

Belirtilen *olasılık* değeri için, ters Cauchy dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileCauchy(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  a,              // dağılımın ortalama parametresi
   const double  b,              // dağılımın ölçek parametresi
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters Cauchy dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileCauchy(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  a,              // dağılımın ortalama parametresi
   const double  b,              // dağılımın ölçek parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters Cauchy dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qcauschy() fonksiyonunun analoğu

```
double  MathQuantileCauchy(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  a,              // dağılımın ortalama parametresi
   const double  b,              // dağılımın ölçek parametresi
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters Cauchy dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileCauchy(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  a,              // dağılımın ortalama parametresi
   const double  b,              // dağılımın ölçek parametresi
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

a

[in] dağılımın ortalama parametresi.

b

[in] dağılımın ölçek parametresi.

tail

[in] Hesaplama bayrağı. 'false' ise hesaplama 1.0 olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomCauchy

a ve b parametrelerini kullanarak, Cauchy dağılımı yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomCauchy(
   const double  a,              // dağılımın ortalama parametresi
   const double  b,              // dağılımın ölçek parametresi
   int&          error_code      // hata kodu değişkeni
   );
```

a ve b parametrelerini kullanarak, Cauchy dağılımı yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekircauchy() fonksiyonunun analoğu

```
bool  MathRandomCauchy(
   const double  a,              // dağılımın ortalama parametresi
   const double  b,              // dağılımın ölçek parametresi
   const int     data_count,     // istenen veri miktarı
   double&       result[]        // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

a

[in] dağılımın ortalama parametresi.

b

[in] dağılımın ölçek parametresi.

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsCauchy

Cauchy dağılımının ilk dört momentinin teorik sayısal değerlerini a ve b parametrelerine göre hesaplar.

```
double  MathMomentsCauchy(
   const double  a,              // dağılımın ortalama parametresi
   const double  b,              // dağılımın ölçek parametresi
   double&       mean,           // ortalama değişkeni
   double&       variance,       // varyans değişkeni
   double&       skewness,       // çarpıklık değişkeni
   double&       kurtosis,       // basıklık değişkeni
   int&          error_code      // hata kodu değişkeni
   );
```

Parametreler

a

[in] dağılımın ortalama parametresi.

b

[in] dağılımın ölçek parametresi.

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

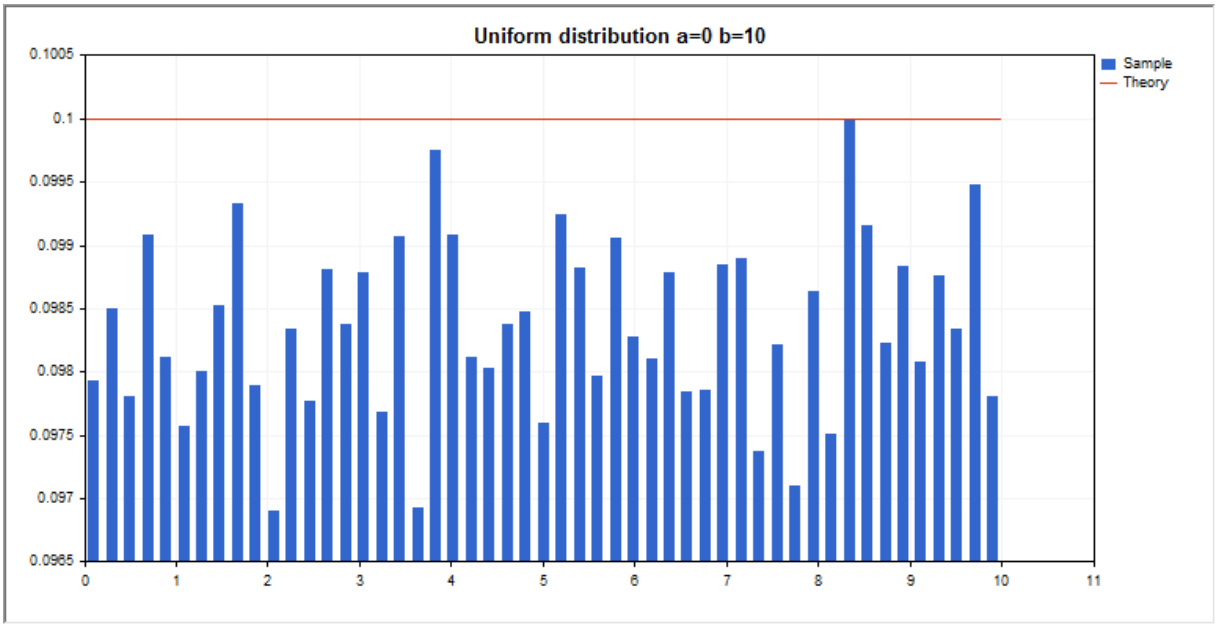
Tekdüze dağılım

Bu bölüm tekdüze dağılım ile çalışmak için tasarlanmış fonksiyonlar içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Tekdüze dağılım şu formülle tanımlanır:

$$f_{Uniform}(x|\ a,b) = \frac{1}{b-a}$$

burada:

- x — rassal değişkenin değeri
- a — dağılım parametresi a (alt sınır)
- b — dağılım parametresi a (üst sınır)



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityUniform	Tekdüze dağılımın olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionUniform	Tekdüze dağılımın olasılık fonksiyonunun değerini hesaplar
MathQuantileUniform	Belli bir olasılık değeri için tekdüze dağılımın olasılık fonksiyonunun değerini hesaplar
MathRandomUniform	Tekdüze dağılım yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur
MathMomentsUniform	Tekdüze dağılımın ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Uniform.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double a_par=0;      // dağılım parametresi a (alt sınır)
input double b_par=10;     // dağılım parametresi b (üst sınır)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // örnekteki değerlerin sayısı
   int ncells=51;       // histogramdaki aralık sayısı
   double x[];          // histogram aralıklarının merkezleri
   double y[];          // aralığın içine düşen değerlerin sayısı
   double data[];       // rassal değişken örneği
   double max,min;      // örnekteki maksimum ve minimum değerlerin sayısı
//--- tekdüze dağılıma uyan bir örnek oluştur
   MathRandomUniform(a_par,b_par,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityUniform(x2,a_par,b_par,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
```

```
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityUniform

Bir rassal x değişkeni için tekdüze dağılımın olasılık yoğunluk fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityUniform(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // dağılım parametresi (alt sınır)
   const double  b,              // dağılım parametresi (üst sınır)
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise o
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için tekdüze dağılımın olasılık yoğunluk fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityUniform(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // dağılım parametresi (alt sınır)
   const double  b,              // dağılım parametresi (üst sınır)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için tekdüze dağılımın olasılık yoğunluk fonksiyonununun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidunif() fonksiyonunun analoğu

```
bool  MathProbabilityDensityUniform(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // dağılım parametresi (alt sınır)
   const double  b,              // dağılım parametresi (üst sınır)
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için tekdüze dağılımın olasılık yoğunluk fonksiyonununun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityUniform(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // dağılım parametresi (alt sınır)
   const double  b,              // dağılım parametresi (üst sınır)
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

a

[in] Dağılım parametresi a (alt sınır).

b

[in] Dağılım parametresi b (üst sınır).

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionUniform

Bir rassal x değişkeni için tekdüze dağılımın olasılık fonksiyonunu a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionUniform(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // dağılım parametresi (alt sınır)
   const double  b,              // dağılım parametresi (üst sınır)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal de
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise ol
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için tekdüze dağılımın olasılık fonksiyonunu a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionUniform(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // dağılım parametresi (alt sınır)
   const double  b,              // dağılım parametresi (üst sınır)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, tekdüze dağılımın olasılık fonksiyonunu a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionUniform(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // dağılım parametresi (alt sınır)
   const double  b,              // dağılım parametresi (üst sınır)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal d
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise d
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, tekdüze dağılımın olasılık fonksiyonunu a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekipunif() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionUniform(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // dağılım parametresi (alt sınır)
   const double  b,              // dağılım parametresi (üst sınır)
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

a

[in] Dağılım parametresi a (alt sınır).

b

[in] Dağılım parametresi b (üst sınır).

tail

[in] hesaplama bayrağı. 'true' ise, olasılık rassal değişkenin x'i aşmayan değerleri için hesaplanır.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerinin dizisi.

MathQuantileUniform

Belirtilen *olasılık* değeri için, ters tekdüze dağılımın olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileUniform(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  a,              // dağılım parametresi (alt sınır)
   const double  b,              // dağılım parametresi (üst sınır)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters tekdüze dağılımın olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileUniform(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  a,              // dağılım parametresi (alt sınır)
   const double  b,              // dağılım parametresi (üst sınır)
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters tekdüze dağılımın olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qcauschy() fonksiyonunun analoğu

```
double  MathQuantileUniform(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  a,              // dağılım parametresi (alt sınır)
   const double  b,              // dağılım parametresi (üst sınır)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters tekdüze dağılımın olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileUniform(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  a,              // dağılım parametresi (alt sınır)
   const double  b,              // dağılım parametresi (üst sınır)
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

a

[in] Dağılım parametresi a (alt sınır).

b

[in] Dağılım parametresi b (üst sınır).

tail

[in] Hesaplama bayrağı. 'false' ise hesaplama 1.0 olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomUniform

a ve b parametrelerini kullanarak, tekdüze dağılım yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomUniform(
   const double  a,              // dağılım parametresi (alt sınır)
   const double  b,              // dağılım parametresi (üst sınır)
   int&          error_code      // hata kodu değişkeni
   );
```

a ve b parametrelerini kullanarak, tekdüze dağılım yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirunif() fonksiyonunun analoğu

```
bool  MathRandomUniform(
   const double  a,              // dağılım parametresi (alt sınır)
   const double  b,              // dağılım parametresi (üst sınır)
   const int     data_count,     // istenen veri miktarı
   double&       result[]        // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

a

[in] Dağılım parametresi a (alt sınır).

b

[in] Dağılım parametresi b (üst sınır).

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsUniform

Tekdüze dağılımın ilk dört momentinin teorik sayısal değerlerini a ve b parametrelerine göre hesaplar.

```
double  MathMomentsUniform(
   const double  a,              // dağılım parametresi (alt sınır)
   const double  b,              // dağılım parametresi (üst sınır)
   double&       mean,           // ortalama değişkeni
   double&       variance,       // varyans değişkeni
   double&       skewness,       // çarpıklık değişkeni
   double&       kurtosis,       // basıklık değişkeni
   int&          error_code      // hata kodu değişkeni
   );
```

Parametreler

a

[in] Dağılım parametresi a (alt sınır).

b

[in] Dağılım parametresi b (üst sınır).

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

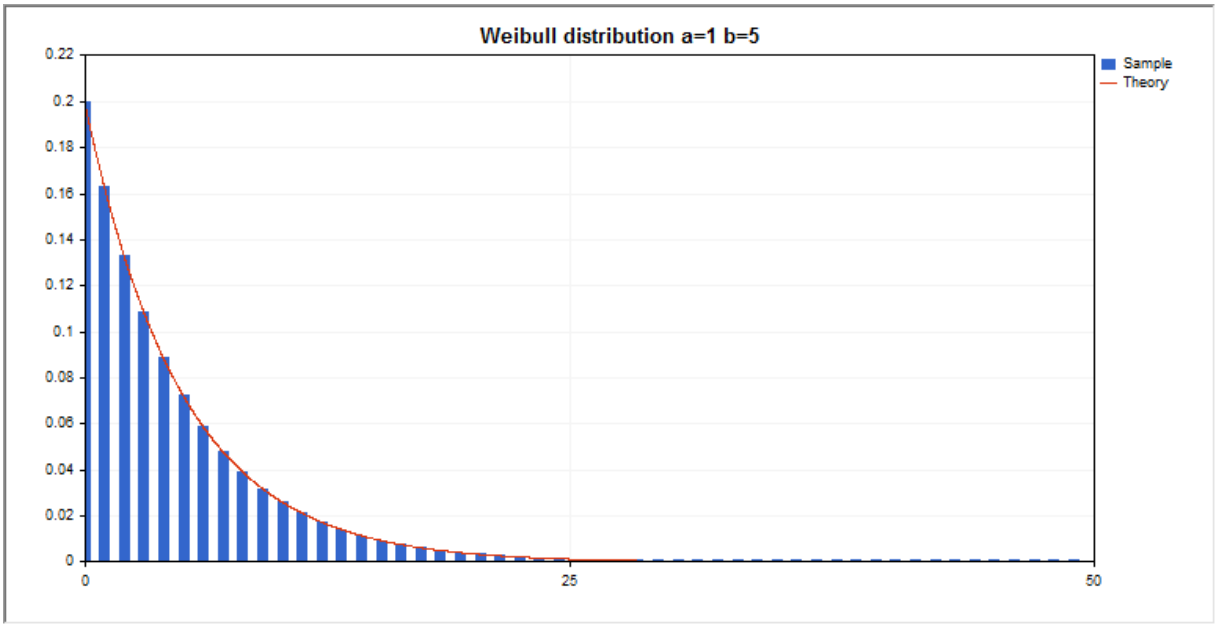
Weibull dağılımı

Bu bölüm Weibull dağılımı ile çalışmak için tasarlanmış fonksiyonlar içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Weibull dağılımı şu formülle tanımlanır:

$$f_{Weibull}(x|a,b) = \frac{a}{b}\left(\frac{x}{b}\right)^{a-1} e^{-\left(\frac{x}{b}\right)^{a}}$$

burada:

- x — rassal değişkenin değeri
- a — dağılım parametresi (şekil)
- b — dağılım parametresi (ölçek)



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityWeibull	Weibull dağılımının olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionWeibull	Weibull dağılımının olasılık fonksiyonunun değerini hesaplar
MathQuantileWeibull	Belli bir olasılık değeri için ters Weibull dağılımının olasılık fonksiyonunun değerini hesaplar
MathRandomWeibull	Weibull dağılımı yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur
MathMomentsWeibull	Weibull dağılımının ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Weibull.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double a_par=1;      // dağılım parametresi (şekil)
input double b_par=5;      // dağılım parametresi (ölçek)
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // örnekteki değerlerin sayısı
   int ncells=51;       // histogramdaki aralık sayısı
   double x[];          // histogram aralıklarının merkezleri
   double y[];          // aralığın içine düşen değerlerin sayısı
   double data[];       // rassal değişken örneği
   double max,min;      // örnekteki maksimum ve minimum değerlerin sayısı
//--- Weibull dağılımına uyan bir örnek oluştur
   MathRandomWeibull(a_par,b_par,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   step=MathMin(step,(max-min)/ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(min,max,step,x2);
   MathProbabilityDensityWeibull(x2,a_par,b_par,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- X ekseninin otomatik olarak ölçeklenmesini engelle
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(max);
   graphic.XAxis().Min(min);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
```

```
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityWeibull

Bir rassal x değişkeni için Weibull dağılımının olasılık yoğunluk fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityWeibull(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // dağılım parametresi (şekil)
   const double  b,              // dağılım parametresi (ölçek)
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise o
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için Weibull dağılımının olasılık yoğunluk fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityWeibull(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // dağılım parametresi (şekil)
   const double  b,              // dağılım parametresi (ölçek)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için Weibull dağılımının olasılık yoğunluk fonksiyonununun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidweibull() fonksiyonunun analoğu

```
bool  MathProbabilityDensityWeibull(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // dağılım parametresi (şekil)
   const double  b,              // dağılım parametresi (ölçek)
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log_
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için Weibull dağılımının olasılık yoğunluk fonksiyonununun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityWeibull(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // dağılım parametresi (şekil)
   const double  b,              // dağılım parametresi (ölçek)
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

a

[in] Dağılım parametresi (ölçek).

b

[in] Dağılım parametresi (şekil).

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionWeibull

Bir rassal x değişkeni için Weibull dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionWeibull(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // dağılım parametresi (şekil)
   const double  b,              // dağılım parametresi (ölçek)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal de
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise ol
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için Weibull dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionWeibull(
   const double  x,              // rassal değişkenin değeri
   const double  a,              // dağılım parametresi (şekil)
   const double  b,              // dağılım parametresi (ölçek)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, Weibull dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekipweibull() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionWeibull(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // dağılım parametresi (şekil)
   const double  b,              // dağılım parametresi (ölçek)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal d
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise d
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, Weibull dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionWeibull(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  a,              // dağılım parametresi (şekil)
   const double  b,              // dağılım parametresi (ölçek)
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

a

[in] Dağılım parametresi (ölçek).

b

[in] Dağılım parametresi (şekil).

tail

[in] hesaplama bayrağı. 'true' ise, olasılık rassal değişkenin x'i aşmayan değerleri için hesaplanır.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerinin dizisi.

MathQuantileWeibull

Belirtilen *olasılık* değeri için, ters Weibull dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileWeibull(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  a,              // dağılım parametresi (şekil)
   const double  b,              // dağılım parametresi (ölçek)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters Weibull dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileWeibull(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  a,              // dağılım parametresi (şekil)
   const double  b,              // dağılım parametresi (ölçek)
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters Weibull dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qweibull() fonksiyonunun analoğu

```
double  MathQuantileWeibull(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  a,              // dağılım parametresi (şekil)
   const double  b,              // dağılım parametresi (ölçek)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters Weibull dağılımının olasılık fonksiyonunun değerini a ve b parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileWeibull(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  a,              // dağılım parametresi (şekil)
   const double  b,              // dağılım parametresi (ölçek)
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

a

[in] Dağılım parametresi (ölçek).

b

[in] Dağılım parametresi (şekil).

tail

[in] Hesaplama bayrağı. 'false' ise hesaplama 1.0 olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomWeibull

a ve b parametrelerini kullanarak, Weibull dağılımı yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomWeibull(
   const double  a,              // dağılım parametresi (şekil)
   const double  b,              // dağılım parametresi (ölçek)
   int&          error_code      // hata kodu değişkeni
   );
```

a ve b parametrelerini kullanarak, Weibull dağılımı yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirweibull() fonksiyonunun analoğu

```
bool  MathRandomWeibull(
   const double  a,              // dağılım parametresi (şekil)
   const double  b,              // dağılım parametresi (ölçek)
   const int     data_count,     // istenen veri miktarı
   double&       result[]        // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

a

[in] Dağılım parametresi (ölçek).

b

[in] Dağılım parametresi (şekil).

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsWeibull

Weibull dağılımının ilk dört momentinin teorik sayısal değerlerini a ve b parametrelerine göre hesaplar.

```
double  MathMomentsWeibull(
   const double  a,              // dağılım parametresi (şekil)
   const double  b,              // dağılım parametresi (ölçek)
   double&       mean,           // ortalama değişkeni
   double&       variance,       // varyans değişkeni
   double&       skewness,       // çarpıklık değişkeni
   double&       kurtosis,       // basıklık değişkeni
   int&          error_code      // hata kodu değişkeni
   );
```

Parametreler

a

[in] Dağılım parametresi (ölçek).

b

[in] Dağılım parametresi (şekil).

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

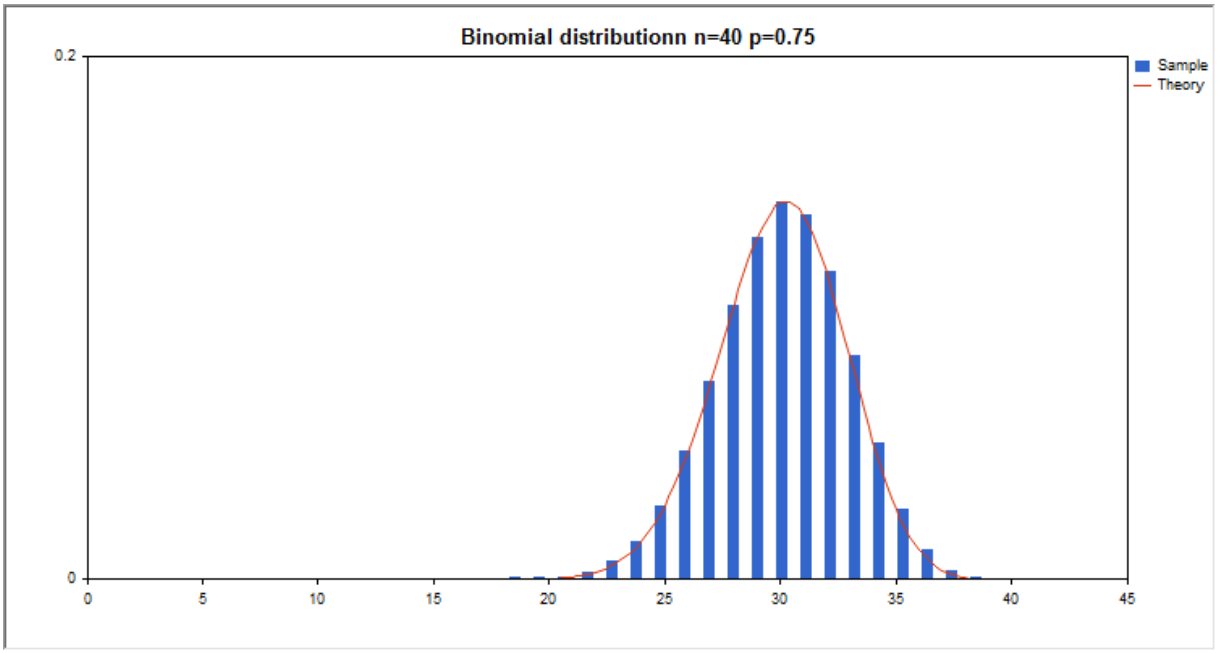
Binomial dağılım

Bu bölüm binomial dağılım ile çalışmak için tasarlanmış fonksiyonlar içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Binomial dağılım şu formülle tanımlanır:

$$f_{Binomial}(x \mid n, p) = \binom{n}{x} p^x (1-p)^{n-x}$$

burada:

- x — rassal değişkenin değeri
- n — deneme sayısı
- p — her bir deneme için başarı olasılığı



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityBinomial	Binomial dağılımın olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionBinomial	Binomial dağılımın olasılık fonksiyonunun değerini hesaplar
MathQuantileBinomial	Belli bir olasılık değeri için ters binomial dağılımın olasılık fonksiyonunun değerini hesaplar
MathRandomBinomial	Binomial dağılım yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur
MathMomentsBinomial	Binomial dağılımın ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Binomial.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double n_par=40;         // deneme sayısı
input double p_par=0.75;       // her bir deney için başarı olasılığı
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;          // örnekteki değerlerin sayısı
   int ncells=20;          // histogramdaki aralık sayısı
   double x[];             // histogram aralıklarının merkezleri
   double y[];             // aralığın içine düşen değerlerin sayısı
   double data[];          // rassal değişken örneği
   double max,min;         // örnekteki maksimum ve minimum değerlerin sayısı
//--- Binom dağılımına uygun bir örnek oluştur
   MathRandomBinomial(n_par,p_par,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityBinomial(x2,n_par,p_par,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   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

Bir rassal x değişkeni için n ve p parametrelerine göre, binomial dağılımın olasılık kütle fonksiyonunun değerini hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityBinomial(
   const double  x,              // rassal değişkenin değeri
   const double  n,              // dağılım parametresi (deney sayısı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçekl
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise ol
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için n ve p parametrelerine göre, binomial dağılımın olasılık kütle fonksiyonunun değerini hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityBinomial(
   const double  x,              // rassal değişkenin değeri
   const double  n,              // dağılım parametresi (deney sayısı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçekl
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için binomial dağılımın olasılık kütle fonksiyonunun değerini n ve p parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidbinom() fonksiyonunun analoğu

```
bool  MathProbabilityDensityBinomial(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  n,              // dağılım parametresi (deney sayısı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log_
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için binomial dağılımın olasılık kütle fonksiyonunun değerini n ve p parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityBinomial(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  n,              // dağılım parametresi (deney sayısı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

n

[in] Dağılım parametresi (deney sayısı).

p

[in] Dağılım parametresi (tek deneme için olayın gerçekleşme olasılığı).

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionBinomial

Bir rassal x değişkeni için, binomial dağılımın olasılık fonksiyonunun değerini n ve p parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionBinomial(
   const double  x,              // rassal değişkenin değeri
   const double  n,              // dağılım parametresi (deney sayısı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçekl
   const bool    tail,           // hesaplama bayrağı. 'true' ise x değerini aşmayan ra
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise ol
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için, binomial dağılımın olasılık fonksiyonunun değerini n ve p parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionBinomial(
   const double  x,              // rassal değişkenin değeri
   const double  n,              // dağılım parametresi (deney sayısı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçekl
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, binomial dağılımın olasılık fonksiyonunun değerini n ve p parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekipweibull() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionBinomial(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  n,              // dağılım parametresi (deney sayısı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   const bool    tail,           // hesaplama bayrağı. 'true' ise x değerini aşmayan r
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise o
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için n ve p parametrelerine göre, binomial dağılımın olasılık fonksiyonunun değerini hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionBinomial(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  n,              // dağılım parametresi (deney sayısı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

n

[in] Dağılım parametresi (deney sayısı).

p

[in] Dağılım parametresi (tek deneme için olayın gerçekleşme olasılığı).

tail

[in] hesaplama bayrağı. 'true' ise, olasılık rassal değişkenin x'i aşmayan değerleri için hesaplanır.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerinin dizisi.

MathQuantileBinomial

Belirtilen *olasılık* değeri için, ters binomial dağılımın olasılık fonksiyonunun değerini n ve p parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileBinomial(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  n,              // dağılım parametresi (deney sayısı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters binomial dağılımın olasılık fonksiyonunun değerini n ve p parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileBinomial(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  n,              // dağılım parametresi (deney sayısı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters binomial dağılımın olasılık fonksiyonunun değerini n ve p parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qbinom() fonksiyonunun analoğu

```
double  MathQuantileBinomial(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  n,              // dağılım parametresi (deney sayısı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters binomial dağılımın olasılık fonksiyonunun değerini n ve p parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileBinomial(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  n,              // dağılım parametresi (deney sayısı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

n

[in] Dağılım parametresi (deney sayısı).

p

[in] Dağılım parametresi (tek deneme için olayın gerçekleşme olasılığı).

tail

[in] Hesaplama bayrağı. 'false' ise hesaplama 1.0 olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomBinomial

n ve p parametrelerini kullanarak, binomial dağılım yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomBinomial(
   const double  n,              // dağılım parametresi (deney sayısı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   int&          error_code      // hata kodu değişkeni
   );
```

n ve p parametrelerini kullanarak, binomial dağılım yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirweibull() fonksiyonunun analoğu

```
bool  MathRandomBinomial(
   const double  n,              // dağılım parametresi (deney sayısı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   const int     data_count,     // istenen veri miktarı
   double&       result[]        // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

n

[in] Dağılım parametresi (deney sayısı).

p

[in] Dağılım parametresi (tek deneme için olayın gerçekleşme olasılığı).

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsBinomial

Binomial dağılımın ilk dört momentinin teorik sayısal değerlerini, n ve p parametrelerine göre hesaplar.

```
double  MathMomentsBinomial(
   const double  n,              // dağılım parametresi (deney sayısı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   double&       mean,           // ortalama değişkeni
   double&       variance,       // varyans değişkeni
   double&       skewness,       // çarpıklık değişkeni
   double&       kurtosis,       // basıklık değişkeni
   int&          error_code      // hata kodu değişkeni
   );
```

Parametreler

n

[in] Dağılım parametresi (deney sayısı).

p

[in] Dağılım parametresi (tek deneme için olayın gerçekleşme olasılığı).

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

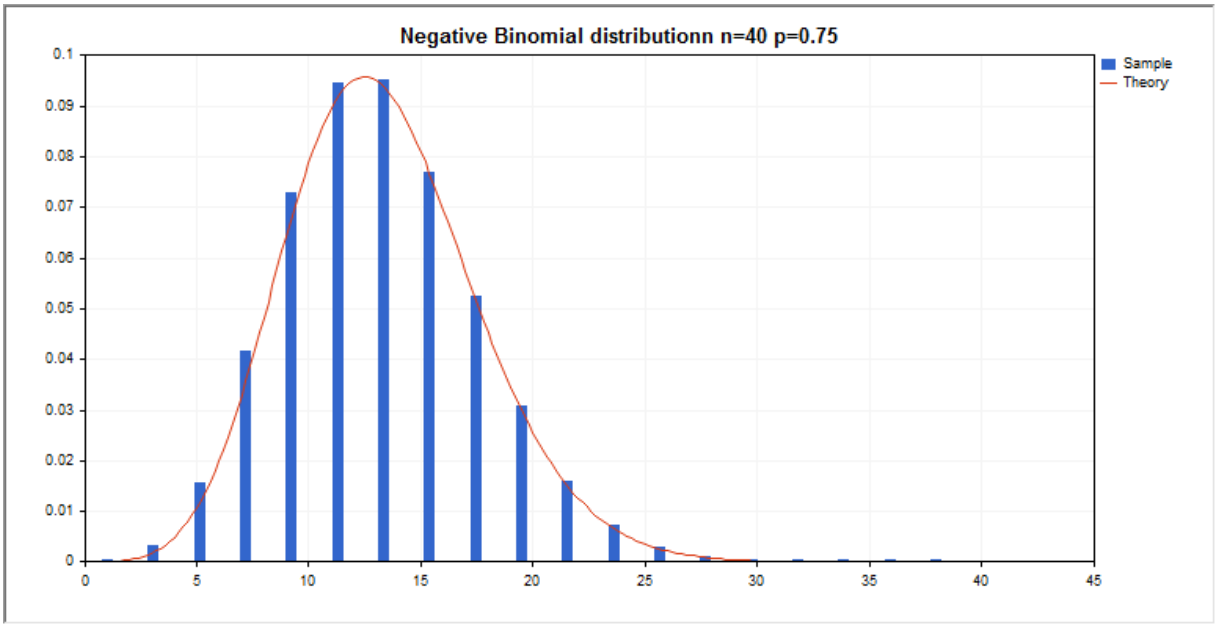
Negatif binomial dağılım

Bu bölüm negatif binomial dağılım ile çalışmak için tasarlanmış fonksiyonlar içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Negatif Binomial dağılım şu formülle tanımlanır:

$$f_{NegativeBinomial}(x \mid r, p) = \frac{\Gamma(r+x)}{\Gamma(r)\Gamma(x+1)} p^r (1-p)^x$$

burada:

- x — rassal değişkenin değeri
- r — başarılı denemelerin sayısı
- p — başarı olasılığı



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityNegativeBinomial	Negatif binomial dağılımın olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionNegativeBinomial	Negatif binomial dağılımın olasılık fonksiyonunun değerini hesaplar
MathQuantileNegativeBinomial	Belli bir olasılık değeri için ters negatif binomial dağılımın olasılık fonksiyonunun değerini hesaplar
MathRandomNegativeBinomial	Negatif binomial dağılım yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur
MathMomentsNegativeBinomial	Negatif binomial dağılımının ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\NegativeBinomial.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double n_par=40;         // deneme sayısı
input double p_par=0.75;       // her bir deney için başarı olasılığı
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;         // örnekteki değerlerin sayısı
   int ncells=19;         // histogramdaki aralık sayısı
   double x[];            // histogram aralıklarının merkezleri
   double y[];            // aralığın içine düşen değerlerin sayısı
   double data[];         // rassal değişken örneği
   double max,min;        // örnekteki maksimum ve minimum değerlerin sayısı
//--- Negatif Binom dağılımına uygun bir örnek oluştur
   MathRandomNegativeBinomial(n_par,p_par,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityNegativeBinomial(x2,n_par,p_par,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   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

Bir rassal x değişkeni için r ve p parametrelerine göre, negatif binomial dağılımın olasılık kütle fonksiyonunun değerini hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityNegativeBinomial(
   const double  x,              // rassal değişkenin değeri (tamsayı)
   const double  r,              // başarılı deneylerin sayısı
   const double  p,              // başarı olasılığı
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise o
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için r ve p parametrelerine göre, negatif binomial dağılımın olasılık kütle fonksiyonunun değerini hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityNegativeBinomial(
   const double  x,              // rassal değişkenin değeri (tamsayı)
   const double  r,              // başarılı deneylerin sayısı
   const double  p,              // başarı olasılığı
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için negatif binomial dağılımın olasılık kütle fonksiyonunun değerini r ve p parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidnbinom() fonksiyonunun analoğu

```
bool  MathProbabilityDensityNegativeBinomial(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  r,              // başarılı deneylerin sayısı
   const double  p,              // başarı olasılığı
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için negatif binomial dağılımın olasılık kütle fonksiyonunun değerini r ve p parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityNegativeBinomial(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  r,              // başarılı deneylerin sayısı
   const double  p,              // başarı olasılığı
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

r

[in] Başarılı denemelerin sayısı

p

[in] Başarı olasılığı.

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionNegativeBinomial

Bir rassal x değişkeni için, negatif binomial dağılımının olasılık fonksiyonunun değerini r ve p parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionNegativeBinomial(
   const double  x,              // rassal değişkenin değeri (tamsayı)
   const double  r,              // başarılı deneylerin sayısı
   const double  p,              // başarı olasılığı
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal de
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise ol
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için, negatif binomial dağılımının olasılık fonksiyonunun değerini r ve p parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionNegativeBinomial(
   const double  x,              // rassal değişkenin değeri (tamsayı)
   const double  r,              // başarılı deneylerin sayısı
   const double  p,              // başarı olasılığı
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, negatif binomial dağılımın olasılık fonksiyonunun değerini r ve p parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekipweibull() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionNegativeBinomial(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  r,              // başarılı deneylerin sayısı
   const double  p,              // başarı olasılığı
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal d
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise d
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, negatif binomial dağılımın olasılık fonksiyonunun değerini r ve p parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionNegativeBinomial(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  r,              // başarılı deneylerin sayısı
   const double  p,              // başarı olasılığı
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

r

[in] Başarılı denemelerin sayısı.

p

[in] Başarı olasılığı.

tail

[in] Hesaplama gecikmesi. 'true' ise x değerini aşmayan rassal değişkenlerin olasılığı hesaplanır

log_mode

[in] Değerin logaritmasının hesaplanma şekli için bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerinin dizisi.

MathQuantileNegativeBinomial

Belirtilen *olasılık* değeri için, ters negatif binomial dağılımın olasılık fonksiyonunun değerini r ve p parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileNegativeBinomial(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  r,              // başarılı deneylerin sayısı
   const double  p,              // başarı olasılığı
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters negatif binomial dağılımın olasılık fonksiyonunun değerini r ve p parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileNegativeBinomial(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  r,              // başarılı deneylerin sayısı
   const double  p,              // başarı olasılığı
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters negatif binomial dağılımın olasılık fonksiyonunun değerini r ve p parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qnbinom() fonksiyonunun analoğu

```
double  MathQuantileNegativeBinomial(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  r,              // başarılı deneylerin sayısı
   const double  p,              // başarı olasılığı
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters negatif binomial dağılımın olasılık fonksiyonunun değerini r ve p parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileNegativeBinomial(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  r,              // başarılı deneylerin sayısı
   const double  p,              // başarı olasılığı
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

r

[in] Başarılı denemelerin sayısı.

p

[in] Başarı olasılığı.

tail

[in] Hesaplama bayrağı. 'false' ise hesaplama 1.0 olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomNegativeBinomial

r ve p parametrelerini kullanarak, negatif binomial dağılım yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomNegativeBinomial(
   const double  r,              // başarılı deneylerin sayısı
   const double  p,              // başarı olasılığı
   int&          error_code      // hata kodu değişkeni
   );
```

r ve p parametrelerini kullanarak, negatif binomial dağılım yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirweibull() fonksiyonunun analoğu

```
bool  MathRandomNegativeBinomial(
   const double  r,              // başarılı deneylerin sayısı
   const double  p,              // başarı olasılığı
   const int     data_count,     // istenen veri miktarı
   double&       result[]        // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

r

[in] Başarılı denemelerin sayısı.

p

[in] Başarı olasılığı.

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsNegativeBinomial

Negatif binomial dağılımın ilk dört momentinin teorik sayısal değerlerini, r ve p parametrelerine göre hesaplar.

```
double  MathMomentsNegativeBinomial(
   const double  r,              // başarılı deneylerin sayısı
   const double  p,              // başarı olasılığı
   double&       mean,           // ortalama değişkeni
   double&       variance,       // varyans değişkeni
   double&       skewness,       // çarpıklık değişkeni
   double&       kurtosis,       // basıklık değişkeni
   int&          error_code      // hata kodu değişkeni
   );
```

Parametreler

r

[in] Başarılı denemelerin sayısı.

p

[in] Başarı olasılığı.

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

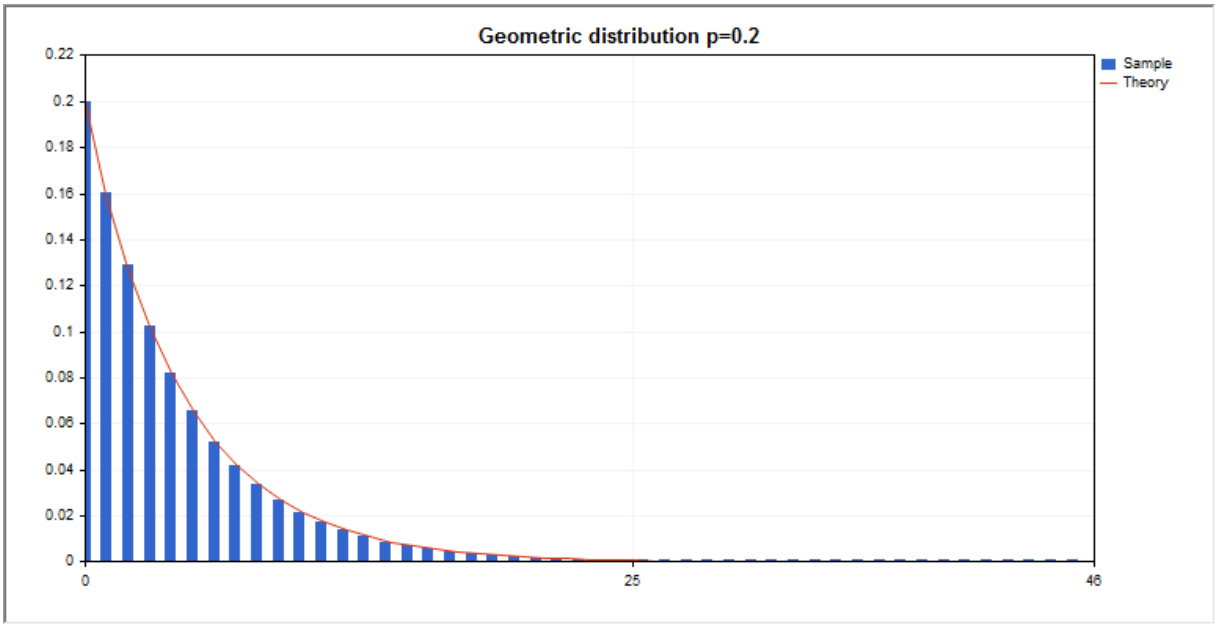
Geometrik dağılım

Bu bölüm geometrik dağılım ile çalışmak için tasarlanmış fonksiyonlar içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Geometrik dağılım şu formülle tanımlanır:

$$f_{Geometric}(x|p) = p(1-p)^x$$

burada:

- x — rassal değişkenin değeri (tamsayı)
- p — bir deneme için olayın gerçekleşme olasılığı



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityGeometric	Geometrik dağılımın olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionGeometric	Geometrik dağılımın olasılık fonksiyonunun değerini hesaplar
MathQuantileGeometric	Belli bir olasılık değeri için ters geometrik dağılımın olasılık fonksiyonunun değerini hesaplar
MathRandomGeometric	geometrik dağılım yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur
MathMomentsGeometric	Geometrik dağılımın ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Geometric.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double p_par=0.2;      // bir deneme için gerçekleşme olasılığı
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // örnekteki değerlerin sayısı
   int ncells=47;       // histogramdaki aralık sayısı
   double x[];          // histogram aralıklarının merkezleri
   double y[];          // aralığın içine düşen değerlerin sayısı
   double data[];       // rassal değişken örneği
   double max,min;      // örnekteki maksimum ve minimum değerlerin sayısı
//--- geometrik dağılıma uyan bir örnek oluştur
   MathRandomGeometric(p_par,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(0,ncells,1,x2);
   MathProbabilityDensityGeometric(x2,p_par,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- X ekseninin otomatik olarak ölçeklenmesini engelle
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Max(max);
   graphic.XAxis().Min(min);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory");
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+i*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
```

```
  {
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityGeometric

Bir rassal x değişkeni için p parametresine göre, geometrik dağılımın olasılık kütle fonksiyonunun değerini hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityGeometric(
   const double  x,              // rassal değişkenin değeri (tamsayı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçekl
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise ol
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için p parametresine göre, geometrik dağılımın olasılık kütle fonksiyonunun değerini hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityGeometric(
   const double  x,              // rassal değişkenin değeri (tamsayı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçekl
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için geometrik dağılımın olasılık kütle fonksiyonunun değerini p parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidgeom() fonksiyonunun analoğu

```
bool  MathProbabilityDensityGeometric(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçekl
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log_
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için geometrik dağılımın olasılık kütle fonksiyonunun değerini p parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityGeometric(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçekl
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

p

[in] Dağılım parametresi (tek deneme için olayın gerçekleşme olasılığı).

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionGeometric

Bir rassal x değişkeni için, geometrik dağılımın olasılık fonksiyonunun değerini p parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionGeometric(
   const double  x,              // rassal değişkenin değeri (tamsayı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçekl
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal de
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise ol
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için, geometrik dağılımın olasılık fonksiyonunun değerini p parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionGeometric(
   const double  x,              // rassal değişkenin değeri (tamsayı)
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçekl
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, geometrik dağılımın olasılık fonksiyonunun değerini p parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekipgeom() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionGeometric(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal d
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise d
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, geometrik dağılımın olasılık fonksiyonunun değerini p parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionGeometric(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   double&       result[]        //olasılık fonksiyonu değerlerinin dizisi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

p

[in] Dağılım parametresi (tek deneme için olayın gerçekleşme olasılığı).

tail

[in] Hesplama bayrağı, 'tail=true' ise, x'i aşmayan rassal değişkenin olasılığı hesaplanır</t5>

log_mode

[in] Değerin logaritmasının hesaplanma şekli için bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık fonksiyonunun değerlerinin dizisi.

MathQuantileGeometric

Belirtilen *olasılık* değeri için, ters geometrik dağılımın olasılık fonksiyonunun değerini p parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileGeometric(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters geometrik dağılımın olasılık fonksiyonunun değerini p parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileGeometric(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters geometrik dağılımın olasılık fonksiyonunun değerini p parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qgeom() fonksiyonunun analoğu

```
double  MathQuantileGeometric(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters geometrik dağılımın olasılık fonksiyonunun değerini p parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileGeometric(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

p

[in] Dağılım parametresi (tek deneme için olayın gerçekleşme olasılığı).

tail

[in] Hesaplama bayrağı. 'false' ise hesaplama 1.0 olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomGeometric

p parametresini kullanarak, geometrik dağılım yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomGeometric(
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   int&          error_code      // hata kodu değişkeni
   );
```

p parametresini kullanarak, geometrik dağılım yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirgeom() fonksiyonunun analoğu

```
bool  MathRandomGeometric(
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   const int     data_count,     // istenen veri miktarı
   double&       result[]        // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

p

[in] Dağılım parametresi (tek deneme için olayın gerçekleşme olasılığı).

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsGeometric

Geometrik dağılımın ilk dört momentinin teorik sayısal değerlerini, p parametresine göre hesaplar.

```
double  MathMomentsGeometric(
   const double  p,              // dağılım parametresi (tek deneme için olayın gerçek
   double&       mean,           // ortalama değişkeni
   double&       variance,       // varyans değişkeni
   double&       skewness,       // çarpıklık değişkeni
   double&       kurtosis,       // basıklık değişkeni
   int&          error_code      // hata kodu değişkeni
   );
```

Parametreler

p

[in] Dağılım parametresi (tek deneme için olayın gerçekleşme olasılığı).

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

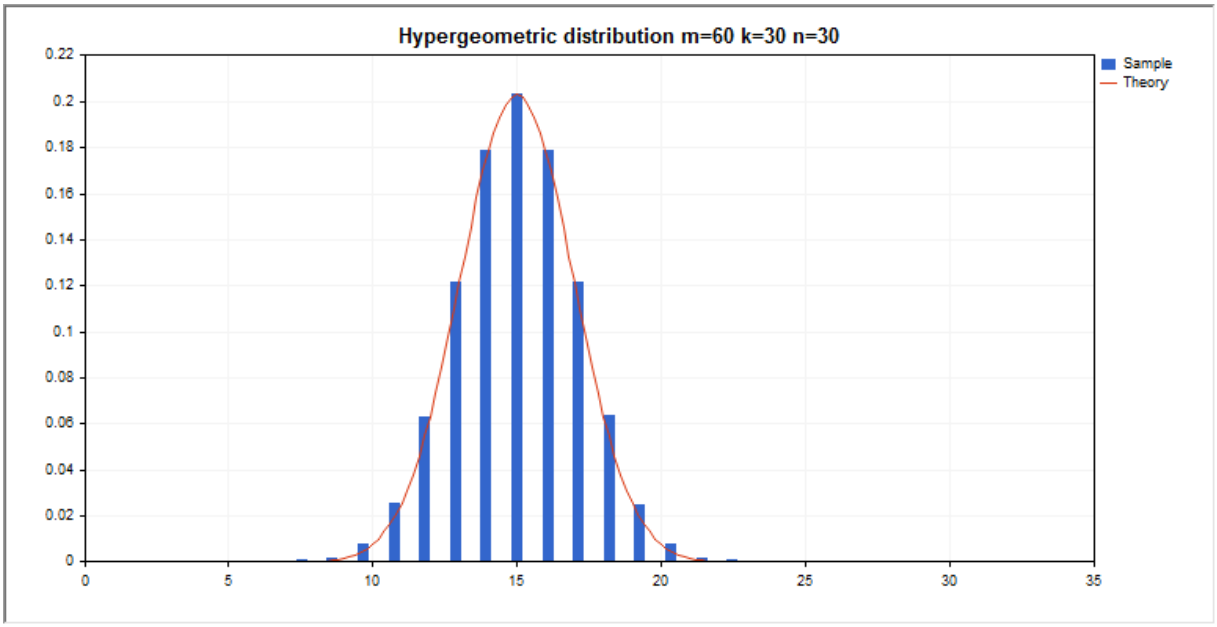
Hipergeometrik dağılım

Bu bölüm hipergeometrik dağılım ile çalışmak için tasarlanmış fonksiyonlar içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Hipergeometrik dağılım şu formülle tanımlanır:

$$f_{Hypergeometric}(x \mid m,k,n) = \frac{\binom{k}{x}\binom{m-k}{n-x}}{\binom{m}{n}}$$

burada:

- x — rassal değişkenin değeri (tamsayı)
- m — toplam nesne sayısı
- k — istenen özelliklere sahip nesnelerin sayısı
- n — çekilen nesnelerin sayısı



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityHypergeometric	Hipergeometrik dağılımın olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionHypergeometric	Hipergeometrik dağılımın olasılık fonksiyonunun değerini hesaplar
MathQuantileHypergeometric	Belli bir olasılık değeri için ters hipergeometrik dağılımın olasılık fonksiyonunun değerini hesaplar

Fonksiyon	Açıklama
MathRandomHypergeometric	Hipergeometrik dağılım yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur
MathMomentsHypergeometric	Hipergeometrik dağılımın ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Hypergeometric.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double m_par=60;      // nesnelerin toplam sayısı
input double k_par=30;      // istenen özelliklere sahip nesnelerin sayısı
input double n_par=30;      // nesne seçimlerinin sayısı
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=1000000;       // örnekteki değerlerin sayısı
   int ncells=15;       // histogramdaki aralık sayısı
   double x[];          // histogram aralıklarının merkezleri
   double y[];          // aralığın içine düşen değerlerin sayısı
   double data[];       // rassal değişken örneği
   double max,min;      // örnekteki maksimum ve minimum değerlerin sayısı
//--- hypergeometrik dağılıma uyan bir örnek oluştur
   MathRandomHypergeometric(m_par,k_par,n_par,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(0,n_par,1,x2);
   MathProbabilityDensityHypergeometric(x2,m_par,k_par,n_par,false,y2);
```

```
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
```

```
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
  {
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityHypergeometric

Bir rassal x değişkeni için, hipergeometrik dağılımın olasılık kütle fonksiyonunun değerini k, m ve n parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityHypergeometric(
   const double  x,              // rassal değişkenin değeri (tamsayı)
   const double  m,              // toplam nesne sayısı (tamsayı)
   const double  k,              // istenen özelliklere sahip olan nesnelerin toplam sa
   const double  n,              // nesne çizimlerinin sayısı (tamsayı)
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise ol
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için, hipergeometrik dağılımın olasılık kütle fonksiyonunun değerini k, m ve n parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityHypergeometric(
   const double  x,              // rassal değişkenin değeri (tamsayı)
   const double  m,              // toplam nesne sayısı (tamsayı)
   const double  k,              // istenen özelliklere sahip olan nesnelerin toplam sa
   const double  n,              // nesne çizimlerinin sayısı (tamsayı)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, hipergeometrik dağılımın olasılık kütle fonksiyonunun değerini k, m ve n parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidhyper() fonksiyonunun analoğu

```
bool  MathProbabilityDensityHypergeometric(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  m,              // toplam nesne sayısı (tamsayı)
   const double  k,              // istenen özelliklere sahip olan nesnelerin toplam s
   const double  n,              // nesne çizimlerinin sayısı (tamsayı)
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log_
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, hipergeometrik dağılımın olasılık kütle fonksiyonunun değerini k, m ve n parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityHypergeometric(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  m,              // toplam nesne sayısı (tamsayı)
   const double  k,              // istenen özelliklere sahip olan nesnelerin toplam s
   const double  n,              // nesne çizimlerinin sayısı (tamsayı)
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

m

[in] Toplam nesne sayısı (tamsayı).

k

[in] İstenen özelliklere sahip olan nesnelerin sayısı (tam sayı).

n

[in] Çekilen nesnelerin sayısı (tamsayı).

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionHypergeometric

Bir rassal x değişkeni için, hipergeometrik dağılımın olasılık fonksiyonunun değerini k, m ve n parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionHypergeometric(
   const double  x,              // rassal değişkenin değeri (tamsayı)
   const double  m,              // toplam nesne sayısı (tamsayı)
   const double  k,              // istenen özelliklere sahip olan nesnelerin toplam sa
   const double  n,              // nesne çizimlerinin sayısı (tamsayı)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal de
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise ol
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için, hipergeometrik dağılımın olasılık fonksiyonunun değerini k, m ve n parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionHypergeometric(
   const double  x,              // rassal değişkenin değeri (tamsayı)
   const double  m,              // toplam nesne sayısı (tamsayı)
   const double  k,              // istenen özelliklere sahip olan nesnelerin toplam sa
   const double  n,              // nesne çizimlerinin sayısı (tamsayı)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, hipergeometrik dağılımın olasılık fonksiyonunun değerini k, m ve n parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidhyper() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionHypergeometric(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  m,              // toplam nesne sayısı (tamsayı)
   const double  k,              // istenen özelliklere sahip olan nesnelerin toplam s
   const double  n,              // nesne çizimlerinin sayısı (tamsayı)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal d
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise d
   double&       result[]        // dağılım fonksiyonu değerlerini içeren dizi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, hipergeometrik dağılımın olasılık fonksiyonunun değerini k, m ve n parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionHypergeometric(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  m,              // toplam nesne sayısı (tamsayı)
   const double  k,              // istenen özelliklere sahip olan nesnelerin toplam s
   const double  n,              // nesne çizimlerinin sayısı (tamsayı)
   double&       result[]        // dağılım fonksiyonu değerlerini içeren dizi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

m

[in] Toplam nesne sayısı (tamsayı).

k

[in] İstenen özelliklere sahip olan nesnelerin sayısı (tam sayı).

n

[in] Çekilen nesnelerin sayısı (tamsayı).

tail

[in] Hesaplama gecikmesi. 'true' ise x değerini aşmayan rassal değişkenlerin olasılığı hesaplanır

log_mode

[in] Değerin logaritmasının hesaplanma şekli için bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Dağılım fonksiyonunun değerleri için kullanılacak dizi.

MathQuantileHypergeometric

Belirtilen *olasılık* değeri için, ters hipergeometrik dağılımın olasılık fonksiyonunun değerini k, m ve n parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileHypergeometric(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  m,              // toplam nesne sayısı (tamsayı)
   const double  k,              // istenen özelliklere sahip olan nesnelerin toplam s
   const double  n,              // nesne çizimlerinin sayısı (tamsayı)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters hipergeometrik dağılımın olasılık fonksiyonunun değerini k, m ve n parametrelerine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantileHypergeometric(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  m,              // toplam nesne sayısı (tamsayı)
   const double  k,              // istenen özelliklere sahip olan nesnelerin toplam s
   const double  n,              // nesne çizimlerinin sayısı (tamsayı)
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters hipergeometrik dağılımın olasılık fonksiyonunun değerini k, m ve n parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qhyper() fonksiyonunun analoğu

```
double  MathQuantileHypergeometric(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  m,              // toplam nesne sayısı (tamsayı)
   const double  k,              // istenen özelliklere sahip olan nesnelerin toplam s
   const double  n,              // nesne çizimlerinin sayısı (tamsayı)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Belirtilen *olasılık* değeri için, ters hipergeometrik dağılımın olasılık fonksiyonunun değerini k, m ve n parametrelerine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantileHypergeometric(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  m,              // toplam nesne sayısı (tamsayı)
   const double  k,              // istenen özelliklere sahip olan nesnelerin toplam s
   const double  n,              // nesne çizimlerinin sayısı (tamsayı)
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

m

[in] Toplam nesne sayısı (tamsayı).

k

[in] İstenen özelliklere sahip olan nesnelerin sayısı (tam sayı).

n

[in] Çekilen nesnelerin sayısı (tamsayı).

tail

[in] Hesaplama bayrağı. lower_tail=false ise hesaplama 1.0-olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomHypergeometric

k, m ve n parametrelerini kullanarak, hipergeometrik dağılım yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomHypergeometric(
   const double  m,              // toplam nesne sayısı (tamsayı)
   const double  k,              // istenen özelliklere sahip olan nesnelerin toplam s
   const double  n,              // nesne çizimlerinin sayısı (tamsayı)
   int&          error_code      // hata kodu değişkeni
   );
```

k, m ve n parametrelerini kullanarak, hipergeometrik dağılım yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirgeom() fonksiyonunun analoğu

```
bool  MathRandomHypergeometric(
   const double  m,              // toplam nesne sayısı (tamsayı)
   const double  k,              // istenen özelliklere sahip olan nesnelerin toplam s
   const double  n,              // nesne çizimlerinin sayısı (tamsayı)
   const int     data_count,     // istenen veri miktarı
   double&       result[]        // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

m

[in] Toplam nesne sayısı (tamsayı).

k

[in] İstenen özelliklere sahip olan nesnelerin sayısı (tam sayı).

n

[in] Çekilen nesnelerin sayısı (tamsayı).

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsHypergeometric

Hipergeometrik dağılımın ilk dört momentinin teorik sayısal değerlerini, k, m ve n parametrelerine göre hesaplar.

```
double  MathMomentsHypergeometric(
   const double  m,              // toplam nesne sayısı (tamsayı)
   const double  k,              // istenen özelliklere sahip olan nesnelerin toplam s
   const double  n,              // nesne çizimlerinin sayısı (tamsayı)
   double&       mean,           // ortalama değişkeni
   double&       variance,       // varyans değişkeni
   double&       skewness,       // çarpıklık değişkeni
   double&       kurtosis,       // basıklık değişkeni
   int&          error_code      // hata kodu değişkeni
   );
```

Parametreler

m

[in] Toplam nesne sayısı (tamsayı).

k

[in] İstenen özelliklere sahip olan nesnelerin sayısı (tam sayı).

n

[in] Çekilen nesnelerin sayısı (tamsayı).

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

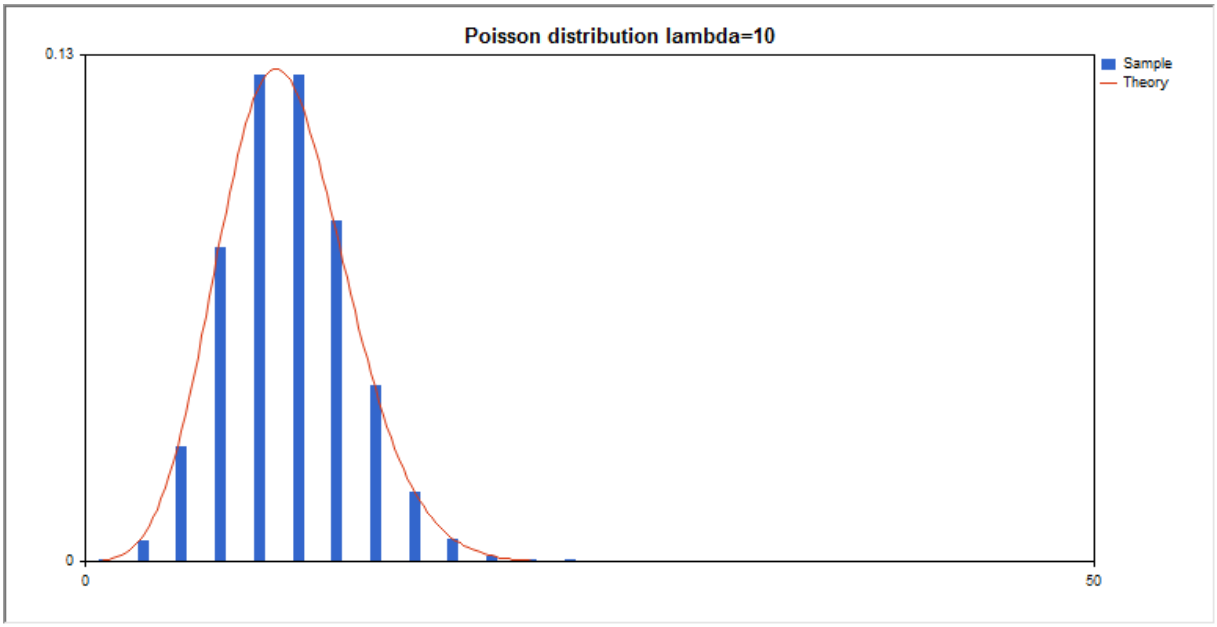
Poisson dağılımı

Bu bölüm Poisson dağılımı ile çalışmak için tasarlanmış fonksiyonlar içerir. Yoğunluğun, olasılığın ve kuantillerin hesaplanmasını ve ilgili yasaya uygun pseudo-rassal sayıların oluşturulmasını sağlar. Poisson dağılımı şu formülle tanımlanır:

$$f_{Poisson}(x|\lambda) = \frac{\lambda^x}{x!}e^{-\lambda}$$

burada:

- x — rassal değişkenin değeri
- λ — dağılım parametresi (ortalama)



Kütüphane, rassal sayıları ayrı ayrı hesaplamanın yanında, rassal sayı dizileriyle çalışmaya da olanak sağlar.

Fonksiyon	Açıklama
MathProbabilityDensityPoisson	Poisson dağılımının olasılık yoğunluk fonksiyonunu hesaplar
MathCumulativeDistributionPoisson	Poisson dağılımının olasılık fonksiyonunun değerini hesaplar
MathQuantilePoisson	Belli bir olasılık değeri için ters Poisson dağılımının olasılık fonksiyonunun değerini hesaplar
MathRandomPoisson	Poisson dağılımı yasasına uygun olarak bir pseudo-rassal değişken veya değişken dizisi oluşturur
MathMomentsPoisson	Poisson dağılımının ilk dört momentinin teorik sayısal değerlerini hesaplar

Örnek:

```
#include <Graphics\Graphic.mqh>
#include <Math\Stat\Poisson.mqh>
#include <Math\Stat\Math.mqh>
#property script_show_inputs
//--- giriş parametreleri
input double lambda_par=10;      // dağılım parametresi ortalama
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- fiyat çizelgesini gizle
   ChartSetInteger(0,CHART_SHOW,false);
//--- rassal sayı üreticisini başlat
   MathSrand(GetTickCount());
//--- rassal değişken örneğini oluştur
   long chart=0;
   string name="GraphicNormal";
   int n=100000;        // örnekteki değerlerin sayısı
   int ncells=13;        // histogramdaki aralık sayısı
   double x[];           // histogram aralıklarının merkezleri
   double y[];           // aralığın içine düşen değerlerin sayısı
   double data[];        // rassal değişken örneği
   double max,min;       // örnekteki maksimum ve minimum değerlerin sayısı
//--- Poisson dağılımına uyan bir örnek oluştur
   MathRandomPoisson(lambda_par,n,data);
//--- histogramı çizmek için gereken verileri hesapla
   CalculateHistogramArray(data,x,y,max,min,ncells);
//--- teorik eğriyi çizebilmek için seri sınırlarını ve adım değerini al
   double step;
   GetMaxMinStepValues(max,min,step);
   PrintFormat("max=%G min=%G",max,min);
//--- hesaplanan teorik verilerden [min,maks] aralığına düşenleri al
   double x2[];
   double y2[];
   MathSequence(0,int(MathCeil(max)),1,x2);
   MathProbabilityDensityPoisson(x2,lambda_par,false,y2);
//--- ölçeği ayarla
   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;
//--- çıktı çizelgeleri
   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);
//--- Y ekseninin otomatik olarak ölçeklenmesini engelle
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Max(NormalizeDouble(theor_max,2));
   graphic.YAxis().Min(0);
//--- tüm eğrileri çiz
   graphic.CurveAdd(x,y,CURVE_HISTOGRAM,"Sample").HistogramWidth(6);
//--- şimdi dağılım yoğunluğunun teorik eğrisini çiz
   graphic.CurveAdd(x2,y2,CURVE_LINES,"Theory").LinesSmooth(true);
   graphic.CurvePlotAll();
//--- tüm eğrileri çiz
   graphic.Update();
  }
//+------------------------------------------------------------------+
//|  Veri setinin frekansını ayarla                                  |
//+------------------------------------------------------------------+
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);
//--- aralık merkezini tanımla
   for(int i=0; i<cells; i++)
     {
      intervals[i]=minv+(i+0.5)*width;
      frequency[i]=0;
     }
//--- aralığın içine düşme frekansını gir
   for(int i=0; i<size; i++)
     {
      int ind=int((data[i]-minv)/width);
      if(ind>=cells) ind=cells-1;
      frequency[ind]++;
     }
   return (true);
  }
//+------------------------------------------------------------------+
//|  Seri oluşturma için gereken değerleri hesaplar                  |
//+------------------------------------------------------------------+
void GetMaxMinStepValues(double &maxv,double &minv,double &stepv)
```

```
  {
//--- normalleştirme kesinliğini almak için serinin tam aralığını hesapla
   double range=MathAbs(maxv-minv);
   int degree=(int)MathRound(MathLog10(range));
//--- maksimum ve minimum değerleri belirtilen kesinlik için normalleştir
   maxv=NormalizeDouble(maxv,degree);
   minv=NormalizeDouble(minv,degree);
//--- seri oluşturma aşaması belirtilen kesinliğe göre ayarlanır
   stepv=NormalizeDouble(MathPow(10,-degree),degree);
   if((maxv-minv)/stepv<10)
      stepv/=10.;
  }
```

MathProbabilityDensityPoisson

Bir rassal x değişkeni için lambda parametresine göre, Poisson dağılımının olasılık kütle fonksiyonunun değerini hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityPoisson(
   const double  x,              // rassal değişkenin değeri (tamsayı)
   const double  lambda,         // dağılım parametresi (ortalama)
   const bool    log_mode,       // değerin logaritmasını hesapla, log_mode=true ise o
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için lambda parametresine göre, Poisson dağılımının olasılık kütle fonksiyonunun değerini hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathProbabilityDensityPoisson(
   const double  x,              // rassal değişkenin değeri (tamsayı)
   const double  lambda,         // dağılım parametresi (ortalama)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, lambda parametresine göre Poisson dağılımının olasılık kütle fonksiyonunun değerini hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidhyper() fonksiyonunun analoğu

```
bool  MathProbabilityDensityPoisson(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  lambda,         // dağılım parametresi (ortalama)
   const bool    log_mode,       // değerin logaritmasını hesaplamak için bayrak, log
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için, lambda parametresine göre Poisson dağılımının olasılık kütle fonksiyonunun değerini hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathProbabilityDensityPoisson(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  lambda,         // dağılım parametresi (ortalama)
   double&       result[]        // olasılık yoğunluk fonksiyonunun değerlerini içeren
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

lambda

[in] Dağılım parametresi (ortalama).

log_mode

[in] Değerin logaritmasını hesaplamak için kullanılan bayrak. log_mode=true ise, olasılık yoğunluk fonksiyonunun doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Olasılık yoğunluk fonksiyonunun değerleri için kullanılacak dizi.

MathCumulativeDistributionPoisson

Bir rassal x değişkeni için Poisson dağılımının olasılık fonksiyonunun değerini lambda parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionPoisson(
   const double  x,              // rassal değişkenin değeri (tamsayı)
   const double  lambda,         // dağılım parametresi (ortalama)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal de
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise ol
   int&          error_code      // hata kodu değişkeni
   );
```

Bir rassal x değişkeni için Poisson dağılımının olasılık fonksiyonunun değerini lambda parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathCumulativeDistributionPoisson(
   const double  x,              // rassal değişkenin değeri (tamsayı)
   const double  lambda,         // dağılım parametresi (ortalama)
   int&          error_code      // hata kodu değişkeni
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için Poisson dağılımının olasılık fonksiyonunun değerini lambda parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindekidhyper() fonksiyonunun analoğu

```
bool  MathCumulativeDistributionPoisson(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  lambda,         // dağılım parametresi (ortalama)
   const bool    tail,           // hesplama bayrağı, 'true' ise, x'i aşmayan rassal d
   const bool    log_mode,       // değerin logaritmasını hesapla. log_mode=true ise d
   double&       result[]        // dağılım fonksiyonu değerlerini içeren dizi
   );
```

Rassal değişkenlerden oluşan bir x[] dizisi için Poisson dağılımının olasılık fonksiyonunun değerini lambda parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathCumulativeDistributionPoisson(
   const double& x[],            // rassal değişkenin değerlerini içeren dizi
   const double  lambda,         // dağılım parametresi (ortalama)
   double&       result[]        // dağılım fonksiyonu değerlerini içeren dizi
   );
```

Parametreler

x

[in] Rassal değişkenin değeri.

x[]

[in] Rassal değişkenin değerlerini içeren dizi.

lambda

[in] Dağılım parametresi (ortalama).

tail

[in] Hesaplama gecikmesi. 'true' ise x değerini aşmayan rassal değişkenlerin olasılığı hesaplanır

log_mode

[in] Değerin logaritmasının hesaplanma şekli için bayrak. log_mode=true ise olasılığın doğal logaritması hesaplanır.

error_code

[out] Hata kodu değişkeni.

result[]

[out] Dağılım fonksiyonunun değerleri için kullanılacak dizi.

MathQuantilePoisson

Belirtilen *olasılık* değeri için, ters Poisson dağılımının olasılık fonksiyonunun değerini lambda parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantilePoisson(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  lambda,         // dağılım parametresi (ortalama)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   int&          error_code      // hata kodu değişkeni
   );
```

Belirtilen *olasılık* değeri için, ters Poisson dağılımının olasılık fonksiyonunun değerini lambda parametresine göre hesaplar. Hata durumunda NaN dönüşü yapar.

```
double  MathQuantilePoisson(
   const double  probability,    // rassal değişkenin gerçekleşme olasılığı
   const double  lambda,         // dağılım parametresi (ortalama)
   int&          error_code      // hata kodu değişkeni
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters Poisson dağılımının olasılık fonksiyonunun değerini lambda parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar. R dilindeki qhyper() fonksiyonunun analoğu

```
double  MathQuantilePoisson(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  lambda,         // dağılım parametresi (ortalama)
   const bool    tail,           // hesaplama bayrağı. 'false' ise hesaplama 1.0-olasi
   const bool    log_mode,       // hesaplama bayrağı. log_mode=true ise hesplama Exp
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Olasılık değerlerinden oluşan *probability[]* dizisi için, ters Poisson dağılımının olasılık fonksiyonunun değerini lambda parametresine göre hesaplar. Hata durumunda 'false' dönüşü yapar.

```
bool  MathQuantilePoisson(
   const double& probability[],  // rassal değişkenin olasılık değerlerini içeren dizi
   const double  lambda,         // dağılım parametresi (ortalama)
   double&       result[]        // kuantil değerlerinin dizisi
   );
```

Parametreler

probability

[in] Rassal değişkenin olasılığı.

probability[]

[in] Rassal değişkenin olasılık değerlerini içeren dizi.

lambda

[in] Dağılım parametresi (ortalama).

tail

[in] Hesaplama bayrağı. lower_tail=false ise hesaplama 1.0-olasılık için yapılır.

log_mode

[in] Hesaplama bayrağı. log_mode=true ise hesaplama Exp(olasılık) için yapılır.

error_code

[out] hata kodunu alacak değişken.

result[]

[out] Kuantil değerlerini içeren dizi.

MathRandomPoisson

lambda parametresini kullanarak, Poisson dağılımı yasasına uygun olarak dağılan bir pseudo-rassal değişken oluşturur. Hata durumunda NaN dönüşü yapar.

```
double  MathRandomPoisson(
   const double  lambda,          // dağılım parametresi (ortalama)
   int&          error_code       // hata kodu değişkeni
   );
```

lambda parametresini kullanarak, Poisson dağılımı yasasına uygun olarak dağılan pseudo-rassal değişkenler oluşturur. Hata durumunda 'false' dönüşü yapar. R dilindekirgeom() fonksiyonunun analoğu

```
bool  MathRandomPoisson(
   const double  lambda,          // dağılım parametresi (ortalama)
   const int     data_count,      // istenen veri miktarı
   double&       result[]         // pseudo-rassal değişkenlerin dizisi
   );
```

Parametreler

lambda

[in] Dağılım parametresi (ortalama).

error_code

[out] Hata kodu değişkeni.

data_count

[out] İstenen veri miktarı.

result[]

[out] Pseudo-rassal değişkenleri içeren dizi.

MathMomentsPoisson

Poisson dağılımının ilk dört momentinin teorik sayısal değerlerini lambda parametresine göre hesaplar.

```
double  MathMomentsPoisson(
   const double  lambda,          // dağılım parametresi (ortalama)
   double&       mean,            // ortalama değişkeni
   double&       variance,        // varyans değişkeni
   double&       skewness,        // çarpıklık değişkeni
   double&       kurtosis,        // basıklık değişkeni
   int&          error_code       // hata kodu değişkeni
   );
```

Parametreler

lambda

[in] Dağılım parametresi (ortalama).

mean

[out] Ortalama değerini alacak değişken.

variance

[out] Varyans değerini alacak değişken.

skewness

[out] Çarpıklık değerini alacak değişken.

kurtosis

[out] Basıklık değerini alacak değişken.

error_code

[out] hata kodunu alacak değişken.

Dönüş Değeri

Momentler başarıyla hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

Alt Fonksiyonlar

Temel matematiksel işlemleri gerçekleştiren fonksiyonlar grubu: gama, beta, faktöriyel, üssel, farklı tabanlarda logaritmik, kare kök, vb. fonksiyonların hesaplanaması.

Bu fonksiyonlar sıradan sayısal değerlerle (reel sayı veya tam sayı) çalışabileceği gibi bu değerlerden oluşan dizilerle de çalışabilir (sonuçlar anı veya ayrı bir diziye yazılır).

Fonksiyon	Açıklama
MathRandomNonZero	0.0 - 1.0 aralığında bir reel tipli rassal sayıya dönüş yapar.
MathMoments	Dizi elemanlarının ilk 4 momentini hesaplar: ortalama, varyans, çarpıklık, basıklık.
MathPowInt	Bir sayının belirtilen üssünü (tamsayı) hesaplar.
MathFactorial	Belirtilen tam sayının faktöriyelini hesaplar.
MathTrunc	Belirtilen dizi elemanlarının tamsayı kısımlarını hesaplar.
MathRound	Belli bir sayıyı veya sayı dizisini belirtilen ondalık basamağa göre yuvarlar.
MathArctan2	[-pi, pi] aralığındaki iki değerin oranı şeklinde bilinen bir tanjant değerinden açı değerini hesaplar.
MathGamma	Gama fonksiyonunun değerini hesaplar.
MathGammaLog	Gama fonksiyonunun logaritmasını hesaplar.
MathBeta	Beta fonksiyonunun değerini hesaplar.
MathBetaLog	Beta fonksiyonunun logaritmasını hesaplar.
MathBetaIncomplete	Tamamlanmamış beta fonksiyonunun değerini hesaplar.
MathGammaIncomplete	Tamamlanmamış gama fonksiyonunun logaritmasını hesaplar.
MathBinomialCoefficient	Binomyal katsayıyı hesaplar.
MathBinomialCoefficientLog	Binomyal katsayının logaritmasını hesaplar.
MathHypergeometric2F2	Hipergeometrik fonksiyonun değerini hesaplar.
MathSequence	İlk eleman, son eleman, artış miktarı değerlerine göre bir seri oluşturur.
MathSequenceByCount	İlk eleman, son eleman, eleman sayısı değerlerine göre bir seri oluşturur.

Fonksiyon	Açıklama
MathReplicate	Tekrarlı değerler serisi oluşturur.
MathReverse	Elemanları ters sıralayarak bir değerler dizisi oluşturur.
MathIdentical	İki diziyi karşılaştırır ve uyan elemanlar için true dönüşü yapar.
MathUnique	Benzersiz değerlerden oluşan bir dizi oluşturur.
MathQuickSortAscending	Dizi elemanlarını artan şekilde sıralar.
MathQuickSortDescending	Dizi elemanlarını azalan şekilde sıralar.
MathQuickSort	Diziyi sıralar.
MathOrder	Elemanların sıralanışına göre permutasyonlu bir dizi oluşturur.
MathBitwiseNot	Dizi elemanları için bitsel NOT (değil) işleminin sonucunu hesaplar.
MathBitwiseAnd	Belirtilen diziler için bitsel AND (ve) işleminin sonucunu hesaplar.
MathBitwiseOr	Belirtilen diziler için bitsel OR (veya) işleminin sonucunu hesaplar.
MathBitwiseXor	Belirtilen diziler için bitsel XOR işleminin sonucunu hesaplar.
MathBitwiseShiftL	Dizi elemanları için bitsel SHL işleminin sonucunu hesaplar.
MathBitwiseShiftR	Dizi elemanları için bitsel SHR işleminin sonucunu hesaplar.
MathCumulativeSum	Birikimli toplamlar dizisi oluşturur.
MathCumulativeProduct	Birikimli çarpımlar dizisi oluşturur.
MathCumulativeMin	Birikimli minimumlu dizi oluşturur.
MathCumulativeMax	Birikimli maksimumlu dizi oluşturur.
MathSin	Dizi elemanları için sin(x) fonksiyonunun değerini hesaplar.
MathCos	Dizi elemanları için cos(x) fonksiyonunun değerini hesaplar.
MathTan	Dizi elemanları için tan(x) fonksiyonunun değerini hesaplar.
MathArcsin	Dizi elemanları için arcsin(x) fonksiyonunun değerini hesaplar.

Fonksiyon	Açıklama
MathArccos	Dizi elemanları için arccos(x) fonksiyonunun değerini hesaplar.
MathArctan	Dizi elemanları için arctan(x) fonksiyonunun değerini hesaplar.
MathSinPi	Dizi elemanları için sin(pi*x) fonksiyonunun değerini hesaplar.
MathCosPi	Dizi elemanları için cos(pi*x) fonksiyonunun değerini hesaplar.
MathTanPi	Dizi elemanları için tan(pi*x) fonksiyonunun değerini hesaplar.
MathAbs	Dizi elemanlarının mutlak değerlerini hesaplar.
MathCeil	Dizi elemanları için en yakın büyük tamsayıya dönüş yapar.
MathFloor	Dizi elemanları için en yakın küçük tamsayıya dönüş yapar.
MathSqrt	Dizi elemanlarının kare köklerini hesaplar.
MathExp	Dizi elemanları için exp(x) fonksiyonunun değerini hesaplar.
MathPow	Dizi elemanları için pow(x,üs) fonksiyonunun değerini hesaplar.
MathLog	Dizi elemanları için log(x) fonksiyonunun değerini hesaplar.
MathLog2	Dizi elemanlarının 2 tabanında logaritmasını hesaplar.
MathLog10	Dizi elemanlarının 10 tabanında logaritmasını hesaplar.
MathDifference	y[i]=x[i+lag]-x[i] farklarına göre bir dizi oluşturur.
MathSample	Dizi elemanlarından rassal bir örnek oluşturur.
MathTukeySummary	Dizi elemanları için Tukey'nin beş-sayılık özetini hesaplar.
MathRange	Dizi elemanlarının maksimum ve minimumlarını hesaplar.
MathMin	Dizi elemanlarının minimumuna dönüş yapar.
MathMax	Dizi elemanlarının maksimumuna dönüş yapar.

Fonksiyon	Açıklama
MathSum	Dizi elemanlarının toplamına dönüş yapar.
MathProduct	Dizi elemanlarının çarpımına dönüş yapar.
MathStandardDeviation	Dizi elemanlarının standart sapmasını hesaplar.
MathAverageDeviation	Dizi elemanlarının mutlak sapmasını hesaplar.
MathMedian	Dizi elemanlarının medyanını hesaplar.
MathMean	Dizi elemanlarının ortalamasını hesaplar.
MathVariance	Dizi elemanlarının varyansını hesaplar.
MathSkewness	Dizi elemanlarının çarpıklığını hesaplar.
MathKurtosis	Dizi elemanlarının basıklığını hesaplar.
MathLog1p	Dizi elemanları için log(1+x) fonksiyonunun değerini hesaplar.
MathExpm1	Dizi elemanları için exp(x)-1 fonksiyonunun değerini hesaplar.
MathSinh	Dizi elemanları için sinh(x) fonksiyonunun değerini hesaplar.
MathCosh	Dizi elemanları için cosh(x) fonksiyonunun değerini hesaplar.
MathTanh	Dizi elemanları için tanh(x) fonksiyonunun değerini hesaplar.
MathArcsinh	Dizi elemanları için arcsinh(x) fonksiyonunun değerini hesaplar.
MathArccosh	Dizi elemanları için arccosh(x) fonksiyonunun değerini hesaplar.
MathArctanh	Dizi elemanları için arctanh(x) fonksiyonunun değerini hesaplar.
MathSignif	Bir değeri, belirtilen ondalık basamak sayısına yuvarlar.
MathRank	Dizi elemanlarının sırasını hesaplar.
MathCorrelationPearson	Pearson korelasyon katsayısını hesaplar.
MathCorrelationSpearman	Spearman korelasyon katsayısını hesaplar.
MathCorrelationKendall	Kendall korelasyon katsayısını hesaplar.
MathQuantile	Belirtilen olasılıklara göre örnek kuantillerini hesaplar.

Fonksiyon	Açıklama
MathProbabilityDensityEmpirical	Rassal değerler için ampirik olasılık yoğunluk fonksiyonunu hesaplar.
MathCumulativeDistributionEmpirical	Rassal değerler için ampirik birikimli dağılım fonksiyonunu hesaplar.

MathRandomNonZero

0.0 - 1.0 aralığında bir reel tipli rassal sayıya dönüş yapar.

```
double  MathRandomNonZero()
```

Dönüş Değeri

0.0 - 1.0 aralığında reel tipli bir rassal sayı.

MathMoments

Dizi elemanlarının ilk 4 momentini hesaplar: ortalama, varyans, çarpıklık, basıklık.

```
bool  MathMoments(
   const double&  array[],            // değerler dizisi
   double&        mean,               // ortalama değişkeni
   double&        variance,           // varyans değişkeni
   double&        skewness,           // çarpıklık değişkeni
   double&        kurtosis,           // basıklık değişkeni
   const int      start=0,            // başlangıç indisi
   const int      count=WHOLE_ARRAY   // eleman sayısı
  )
```

Parametreler

array[]

[in] Değerler dizisi.

mean

[out] Ortalama değişkeni (birinci moment).

variance

[out] Varyans değişkeni (ikinci moment).

skewness

[out] Çarpıklık değişkeni (üçüncü moment).

kurtosis

[out] Basıklık değişkeni (dördüncü moment).

start=0

[in] Hesaplama için başlangıç indisi.

count=WHOLE_ARRAY

[in] Hesaplamaya katılacak elemanların sayısı.

Dönüş Değeri

Momentler başarılı şekilde hesaplanmışsa 'true', aksi durumda 'false' dönüşü yapar.

MathPowInt

Bir sayının belirtilen üssünü (tamsayı) hesaplar.

```
double  MathPowInt(
   const double  x,       // sayının değeri
   const int     power    // üs değeri
  )
```

Parametreler

x

[in] Üssü hesaplanacak reel değerli (çift duyarlı) sayı.

power

[in] Üs değeri (tamsayı).

Dönüş Değeri

x sayısının belirtilen üssünün değeri.

MathFactorial

Belirtilen tam sayının faktöriyelini hesaplar.

```
double  MathFactorial(
   const int  n   // sayının değeri
  )
```

Parametreler

n

[in] Faktöriyeli hesaplanacak tamsayı.

Dönüş Değeri

Sayının faktöriyeli.

MathTrunc

Belirtilen dizi elemanlarının tamsayı kısımlarını hesaplar.

Kayan noktalı çift duyarlık sayısı ile çalışan versiyon:

```
double  MathTrunc(
   const double   x          // sayının değeri
  )
```

Dönüş Değeri

Belirtilen sayının tam kısmı.

Kayan noktalı çift duyarlıklı sayı dizisiyle çalışan versiyon: Sonuçlar yeni bir diziye yazılır:

```
bool  MathTrunc(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Kayan noktalı çift duyarlıklı sayı dizisiyle çalışan versiyon: Sonuçlar girdi dizisinin üstüne yazılır:

```
bool  MathTrunc(
   double&        array[]    // değerler dizisi
  )
```

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Parametreler

x

[in] Tam kısmı alınacak olan kayan noktalı çift duyarlıklı sayı.

array[]

[in] Tam kısımları alınacak olan kayan noktalı çift duyarlıklı sayılardan oluşan dizi.

array[]

[out] Sonuç değerlerinin dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

MathRound

Çifte-duyarlıklı kayan noktalı bir sayıyı veya diziyi, belirtilen ondalık basamak sayısına yuvarlar.

Çifte-duyarlıklı kayan noktalı sayının, belirtilen ondalık basamak sayısına yuvarlandığı versiyon:

```
double  MathRound(
   const double   x,          // sayının değeri
   const int      digits      // ondalık basamakların sayısı
  )
```

Dönüş Değeri

Ondalık basamakları 'digits' parametresi kadar olan, 'x' parametresine en yakın sayı.

Çifte-duyarlıklı kayan noktalı sayılar dizisinin, belirtilen ondalık basamak sayısına yuvarlandığı versiyon: Sonuçlar yeni bir diziye yazılır.

```
bool  MathRound(
   const double&  array[],    // değerler dizisi
   int            digits,     // ondalık basamakların sayısı
   double&        result[]    // sonuçlar dizisi
  )
```

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Çifte-duyarlıklı kayan noktalı sayılar dizisinin, belirtilen ondalık basamak sayısına yuvarlandığı versiyon: Sonuçlar girdi dizisinin üstüne yazılır.

```
bool  MathRound(
   double&        array[],    // değerler dizisi
   int            digits      // ondalık basamakların sayısı
  )
```

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Parametreler

x

[in] Yuvarlanacak olan kayan noktalı çift duyarlıklı sayı.

digits

[in] Dönüş değerinin ondalık basamak sayısı.

array[]

[in] Yuvarlanacak olan kayan noktalı çift duyarlıklı sayı dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

MathArctan2

(x, y) şeklindeki iki argümanın oranının ters tanjantına (ilgili açının radyan değerine) dönüş yapar.

Belirtilen (x, y) sayılarının oranıyla çalışan versiyon:

```
double  MathArctan2(
   const double       y,          // Y koordinatı
   const double       x           // X koordinatı
  )
```

Dönüş Değeri

-π≤θ≤π aralığında radyan cinsinden θ açısının değeri (burada tan (θ) = y veya x ve (x, y) Kartezyen koordinat sisteminde bir nokta).

Belirtilen (x, y) sayılarının oranlarından olşan bir diziyle çalışan versiyon:

```
bool  MathArctan2(
   const double&      x[],        // x değerlerinin dizisi
   const double&      y[],        // y değerlerinin dizisi
   double&            result[]    // sonuçlar dizisi
  )
```

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Parametreler

y

[in] Noktanın Y koordinatı.

x

[in] Noktanın X koordinatı.

x[]

[in] Noktaların X koordinatlarını içeren dizi.

y[]

[in] Noktaların Y koordinatlarını içeren dizi.

result[]

[out] Sonuçların yazılacağı dizi

Notlar

Lütfen şunları not alın.

- Birinci bölgedeki (x, y) noktası için, dönüş değeri (0, π/2) aralığında olacaktır.
- İkinci bölgedeki (x, y) noktası için, dönüş değeri (π/2,π] aralığında olacaktır.
- Üçüncü bölgedeki (x, y) noktası için, dönüş değeri (-π,π/2) aralığında olacaktır.
- Dördüncü bölgedeki (x, y) noktası için, dönüş değeri (-π/2,0) aralığında olacaktır.

Bu bölgeler dışında kalan noktalar için dönüş değeri şu şekilde olur.

- y = 0 ve x > 0 ise, θ = 0.
- y = 0 ve x < 0 ise, θ = π.
- x = 0 ve y > 0 ise, θ = π/2.
- x = 0 ve y < 0 ise, θ = -π/2.
- x = 0 ve y = 0 ise, θ = -π/2.

x veya y değerleri, NaN (tanımsız) veya PositiveInfinity (+ sonsuz) veya NegativeInfinity (- sonsuz) değerlerine sahipse, fonksiyon NaN değerine dönüş yapar.

MathGamma

Gama fonksiyonunun değerini *x* reel argümanı için hespalar.

```
double  MathGamma(
   const double  x       // fonksiyon argümanı
  )
```

Parametreler

x

[in] Fonksiyonun yerel argümanı.

Dönüş Değeri

Gama fonnksiyonunun değeri.

MathGammaLog

Gama fonksiyonunun logaritmasını x reel argümanı için hespalar.

```
double  MathGammaLog(
   const double  x      // fonksiyon argümanı
  )
```

Parametreler

x

[in] Fonksiyonun yerel argümanı.

Dönüş Değeri

Fonksiyonun logaritması.

MathBeta

Beta fonksiyonunun değerini a ve b reel argümanları için hespalar.

```
double  MathBeta(
   const double  a,       // fonksiyonun ilk argümanı
   const double  b        // fonksiyonun ikinci argümanı
  )
```

Parametreler

a

[in] Fonksiyonun a argümanı

b

[in] Fonksiyonun b argümanı

Dönüş Değeri

Fonksiyon değeri.

MathBetaLog

Beta fonksiyonunun logaritmasını a ve b reel argümanları için hespalar.

```
double  MathBetaLog(
   const double  a,       // fonksiyonun ilk argümanı
   const double  b        // fonksiyonun ikinci argümanı
  )
```

Parametreler

a

[in] Fonksiyonun a argümanı

b

[in] Fonksiyonun b argümanı

Dönüş Değeri

Fonksiyonun logaritması.

MathBetaIncomplete

Tamamlanmamış beta fonksiyonunun değerini hesaplar.

```
double  MathBetaIncomplete(
   const double  x,       // fonksiyon argümanı
   const double  p,       // fonksiyonun ilk parametresi
   const double  q        // fonksiyonun ikinci parametresi
  )
```

Parametreler

x

[in] Fonksiyonun argümanı.

p

[in] Beta fonksiyonunun ilk parametresi, 0.0 'dan büyük olmalı.

q

[in] Beta fonksiyonunun ikinci parametresi, 0.0 'dan büyük olmalı.

Dönüş Değeri

Fonksiyon değeri.

MathGammaIncomplete

Tamamlanmamış gama fonksiyonunun logaritmasını hesaplar..

```
double  MathGammaIncomplete(
   double  x,           // fonksiyon argümanı
   double  alpha        // fonksiyon parametresi
  )
```

Parametreler

x

[in] Fonksiyonun argümanı.

alpha

[in] Tamamlanmamış gama fonksiyonunun parametresi.

Dönüş Değeri

Fonksiyon değeri.

MathBinomialCoefficient

Binamyal katsayıyı hesaplar: C(n,k)=n!/(k!*(n-k)!).

```
long  MathBinomialCoefficient(
   const int  n,       //
   const int  k        // kombinasyondaki elemanların toplam sayısı
  )
```

Parametreler

n

[in] Elemanların sayısı.

k

[in] Her bir kombinasyondaki eleman sayısı.

Dönüş Değeri

N'in K'lı kombinasyonlarının sayısı.

MathBinomialCoefficientLog

Binamyal katsayının logaritmasını hesaplar: Log(C(n,k))=Log(n!/(k!*(n-k)!))

Tamsayı argümanlar için:

```
double  MathBinomialCoefficientLog(
   const int      n,       //
   const int      k        // kombinasyondaki elemanların toplam sayısı
  );
```

Reel argümanlar için:

```
double  MathBinomialCoefficientLog(
   const double   n,       // elemanların toplam sayısı
   const double   k        // kombinasyondaki elemanların toplam sayısı
  );
```

Parametreler

n

[in] Elemanların sayısı.

k

[in] Her bir kombinasyondaki eleman sayısı.

Dönüş Değeri

C(n,k) değerinin logaritması.

MathHypergeometric2F2

Hypergeometric_2F2 (a, b, c, d, z) fonksiyonunun değerinin Taylor yöntemiyle hesaplar.

```
double  MathHypergeometric2F2(
   const double  a,      // fonksiyonun ilk parametresi
   const double  b,      // fonksiyonun ikinci parametresi
   const double  c,      // fonksiyonun üçüncü parametresi
   const double  d,      // fonksiyonun dördüncü parametresi
   const double  z       // fonksiyonun beşinci parametresi
  )
```

Parametreler

a

[in] Fonksiyonun ilk parametresi.

b

[in] Fonksiyonun ikinci parametresi.

c

[in] Fonksiyonun üçüncü parametresi.

d

[in] Fonksiyonun dördüncü parametresi. .

z

[in] Fonksiyonun beşinci parametresi. .

Dönüş Değeri

Fonksiyon değeri.

MathSequence

İlk eleman, son eleman, artış miktarı değerlerine göre bir seri oluşturur.

Reel değerler için kullanılan versiyon:

```
bool  MathSequence(
   const double  from,        // başlangıç değeri
   const double  to,          // son değer
   const double  step,        // adım
   double&       result[]     // sonuçlar dizisi
  )
```

Tamsayı değerler için kullanılan versiyon:

```
bool  MathSequence(
   const int     from,        // başlangıç değeri
   const int     to,          // son değer
   const int     step,        // adım
   int&          result[]     // sonuçlar dizisi
  )
```

Parametreler

from

[in] Serinin ilk değeri

to

[in] Serinin son değeri

step

[in] Serinin ilerleme değeri.

result[]

[out] Serinin yazılacağı dizi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathSequenceByCount

İlk eleman, son eleman, eleman sayısı değerlerine göre bir seri oluşturur.

Reel değerler için kullanılan versiyon:

```
bool  MathSequenceByCount(
   const double  from,        // başlangıç değeri
   const double  to,          // son değer
   const int     count,       // sayı
   double&       result[]     // sonuçlar dizisi
  )
```

Tamsayı değerler için kullanılan versiyon:

```
bool  MathSequenceByCount(
   const int     from,        // başlangıç değeri
   const int     to,          // son değer
   const int     count,       // sayı
   int&          result[]     // sonuçlar dizisi
  )
```

Parametreler

from

[in] Serinin ilk değeri.

to

[in] Serinin son değeri.

count

[in] Serideki eleman sayısı.

result[]

[out] Serinin yazılacağı dizi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathReplicate

Tekrarlı değerler serisi oluşturur.

Reel değerler için kullanılan versiyon:

```
bool  MathReplicate(
   const double&  array[],    // değerler dizisi
   const int      count,      // tekrar sayısı
   double&        result[]    // sonuçlar dizisi
  )
```

Tamsayı değerler için kullanılan versiyon:

```
bool  MathReplicate(
   const int&     array[],    // değerler dizisi
   const int      count,      // tekrar sayısı
   int&           result[]    // sonuçlar dizisi
  )
```

Parametreler

array[]

[in] Seri oluşturmak için kullanılacak dizi.

count

[in] Serideki tekrarların sayısı.

result[]

[out] Serinin yazılacağı dizi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathReverse

Elemanları ters sıralayarak bir değerler dizisi oluşturur.

Reel değerlerle çalışan ve sonuçları yeni bir diziye yazan versiyon.

```
bool  MathReverse(
   const double&  array[],      // değerler dizisi
   double&        result[]      // sonuçlar dizisi
  )
```

Tamsayı değerlerle çalışan ve sonuçları yeni bir diziye yazan versiyon.

```
bool  MathReverse(
   const int&     array[],      // değerler dizisi
   int&           result[]      // sonuçlar dizisi
  )
```

Reel değerlerle çalışan ve sonuçları aynı dizinin üstüne yazan versiyon.

```
bool  MathReverse(
   double&        array[]       // değerler dizisi
  )
```

Tamsayı değerlerle çalışan ve sonuçları aynı dizinin üstüne yazan versiyon.

```
bool  MathReverse(
   int&           array[]       // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

array[]

[out] Değerlerin ters sıralandığı sonuç dizisi.

result[]

[out] Değerlerin ters sıralandığı sonuç dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathIdentical

İki diziyi karşılaştırır ve uyan elemanlar için true dönüşü yapar.

Reel değerli diziler için kullanılan versiyon:

```
bool  MathIdentical(
   const double&  array1[],      // birinci değerler dizisi
   const double&  array2[]       // ikinci değerler dizisi
  )
```

Tamsayı değerli diziler için kullanılan versiyon:

```
bool  MathIdentical(
   const int&     array1[],      // birinci değerler dizisi
   const int&     array2[]       // ikinci değerler dizisi
  )
```

Parametreler

array1[]

[in] Karşılaştırılacak ilk dizi.

array2[]

[in] Karşılaştırılacak ikinci dizi.

Dönüş Değeri

Diziler eşit ise 'true', aksi durumda 'false'.

MathUnique

Benzersiz değerlerden oluşan bir dizi oluşturur.

Reel değerler için kullanılan versiyon:

```
bool  MathUnique(
   const double&   array[],   // değerler dizisi
   double&         result[]   // sonuçlar dizisi
  )
```

Tamsayı değerler için kullanılan versiyon:

```
bool  MathUnique(
   const int&     array[],      // değerler dizisi
   int&           result[]      // sonuçlar dizisi
  )
```

Parametreler

array[]

[in] Kaynak dizisi.

result[]

[out] Benzersiz değerlerin girileceği çıktı dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathQuickSortAscending

QuickSort (hızlı sıralama) algoritmasını kullanarak array[] ve indices[] dizilerini eş zamanlı olarak artan sırayla sıralar.

```
void  MathQuickSortAscending(
   double&  array[],      // değerler dizisi
   int&     indices[],    // indisler dizisi
   int      first,        // başlangıç değeri
   int      last          // son değer
  )
```

Parametreler

array[]

[in][out] Sıralanacak dizi.

indices[]

[in][out] Orjinal dizinin indislerinin yazılacağı dizi.

first

[in] Sıralamanın başlatılacağı elemanın indisi.

last

[in] Sıralamanın sonlandırılacağı elemanın indisi.

MathQuickSortDescending

QuickSort (hızlı sıralama) algoritmasını kullanarak array[] ve indices[] dizilerini eş zamanlı olarak azalan sırayla sıralar.

```
void  MathQuickSortDescending(
   double&  array[],      // değerler dizisi
   int&     indices[],    // indisler dizisi
   int      first,        // başlangıç değeri
   int      last          // son değer
  )
```

Parametreler

array[]

[in][out] Sıralanacak dizi.

indices[]

[in][out] Orjinal dizinin indislerinin yazılacağı dizi.

first

[in] Sıralamanın başlatılacağı elemanın indisi.

last

[in] Sıralamanın sonlandırılacağı elemanın indisi.

MathQuickSort

QuickSort (hızlı sıralama) algoritmasını kullanarak array[] ve indices[] dizilerini eş zamanlı olarak sıralar.

```
void  MathQuickSort(
   double&  array[],      // değerler dizisi
   int&     indices[],    // indisler dizisi
   int      first,        // başlangıç değeri
   int      last,         // son değer
   int      mode          // yön
  )
```

Parametreler

array[]

[in][out] Sıralanacak dizi.

indices[]

[in][out] Orjinal dizinin indislerinin yazılacağı dizi.

first

[in] Sıralamanın başlatılacağı elemanın indisi.

last

[in] Sıralamanın sonlandırılacağı elemanın indisi.

mode

[in] Sıralama yönü (>0 artan; aksi durumda, azalan).

MathOrder

Sıralama işleminin ardından elemanların sıralanışına göre permütasyonlu bir dizi oluşturur.

Reel değerli diziler için kullanılan versiyon:

```
bool  MathOrder(
   const double&  array[],   // değerler dizisi
   int&           result[]   // sonuçlar dizisi
  )
```

Tamsayı değerli diziler için kullanılan versiyon:

```
bool  MathOrder(
   const int&     array[],   // değerler dizisi
   int&           result[]   // sonuçlar dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sıralanan indislerin yazılacağı dizi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathBitwiseNot

Dizi elemanları için bitsel NOT (değil) işleminin sonucunu hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathBitwiseNot(
   const int&  array[],   // değerler dizisi
   int&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathBitwiseNot(
   int&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathBitwiseAnd

Belirtilen diziler için bitsel AND işleminin sonucunu hesaplar.

```
bool  MathBitwiseAnd(
   const int&  array1[],  // birinci değerler dizisi
   const int&  array2[],  // ikinci değerler dizisi
   int&        result[]   // sonuçlar dizisi
  )
```

Parametreler

array1[]

[in] The first Değerler dizisi.

array2[]

[in] The second Değerler dizisi.

result[]

[out] Sonuçların yazılacağı dizi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathBitwiseOr

Belirtilen diziler için bitsel OR işleminin sonucunu hesaplar.

```
bool  MathBitwiseOr(
   const int&  array1[],  // birinci değerler dizisi
   const int&  array2[],  // ikinci değerler dizisi
   int&        result[]   // sonuçlar dizisi
  )
```

Parametreler

array1[]

[in] The first Değerler dizisi.

array2[]

[in] The second Değerler dizisi.

result[]

[out] Sonuçların yazılacağı dizi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathBitwiseXor

Belirtilen diziler için bitsel XOR işleminin sonucunu hesaplar.

```
bool  MathBitwiseXor(
   const int&  array1[],  // birinci değerler dizisi
   const int&  array2[],  // ikinci değerler dizisi
   int&        result[]   // sonuçlar dizisi
  )
```

Parametreler

array1[]

[in] The first Değerler dizisi.

array2[]

[in] The second Değerler dizisi.

result[]

[out] Sonuçların yazılacağı dizi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathBitwiseShiftL

Dizi elemanları için bitsel SHL (sola kaydırma) işleminin sonucunu hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathBitwiseShiftL(
   const int&  array[],    // değerler dizisi
   const int   n,          // kaydırma değeri
   int&        result[]    // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathBitwiseShiftL(
   int&        array[],    // değerler dizisi
   const int   n           // kaydırma değeri
  )
```

Parametreler

array[]

[in] Değerler dizisi.

n

[in] Kaydırılacak bit sayısı.

array[]

[out] Sonuç değerlerinin dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathBitwiseShiftR

Dizi elemanları için bitsel SHR (sağa kaydırma) işleminin sonucunu hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathBitwiseShiftR(
   const int&  array[],    // değerler dizisi
   const int   n,          // kaydırma değeri
   int&        result[]    // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathBitwiseShiftR(
   int&        array[],    // değerler dizisi
   const int   n           // kaydırma değeri
  )
```

Parametreler

array[]

[in] Değerler dizisi.

n

[in] Kaydırılacak bit sayısı.

array[]

[out] Sonuç değerlerinin dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathCumulativeSum

Birikimli toplamlar dizisi oluşturur.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathCumulativeSum(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathCumulativeSum(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathCumulativeProduct

Birikimli çarpımlar dizisi oluşturur.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathCumulativeProduct(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathCumulativeProduct(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathCumulativeMin

Birikimli minimumlu dizi oluşturur.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathCumulativeMin(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathCumulativeMin(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathCumulativeMax

Birikimli maksimumlu dizi oluşturur.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathCumulativeMax(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathCumulativeMax(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathSin

Dizi elemanları için sin(x) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathSin(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathSin(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathCos

Dizi elemanları için cos(x) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathCos(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathCos(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathTan

Dizi elemanları için tan(x) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathTan(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathTan(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathArcsin

Dizi elemanları için arcsin(x) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathArcsin(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathArcsin(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathArccos

Dizi elemanları için arccos(x) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathArccos(
   const double&   array[],    // değerler dizisi
   double&         result[]    // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathArccos(
   double&         array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathArctan

Dizi elemanları için arctan(x) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathArctan(
   const double&   array[],   // değerler dizisi
   double&         result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathArctan(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathSinPi

Dizi elemanları için sin(pi*x) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathSinPi(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathSinPi(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathCosPi

Dizi elemanları için cos(pi*x) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathCosPi(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathCosPi(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathTanPi

Dizi elemanları için tan(pi*x) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathTanPi(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathTanPi(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathAbs

Dizi elemanlarının mutlak değerlerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathAbs(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathAbs(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathCeil

Dizi elemanları için en yakın büyük tamsayıya dönüş yapar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathCeil(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathCeil(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathFloor

Dizi elemanları için en yakın küçük tamsayıya dönüş yapar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathFloor(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathFloor(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathSqrt

Dizi elemanlarının kare köklerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathSqrt(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathSqrt(
   double&         array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathExp

Dizi elemanları için exp(x) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathExp(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathExp(
   double&       array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathPow

Dizi elemanları için pow(x,üs) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathPow(
   const double&  array[],    // değerler dizisi
   const double   power,      // üs
   double&        result[]    // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathPow(
   double&        array[],    // değerler dizisi
   const double   power       // üs
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathLog

Dizi elemanları için log(x) fonksiyonunun değerini hesaplar.

Doğal logaritmayı hesaplayıp sonuçları yeni bir diziye yazan versiyon.

```
bool  MathLog(
   const double&  array[],      // değerler dizisi
   double&        result[]      // sonuçlar dizisi
  )
```

Doğal logaritmayı hesaplayıp sonuçları girdi dizisinin üstüne yazan versiyon.

```
bool  MathLog(
   double&        array[]       // değerler dizisi
  )
```

Belli bir tabana göre logaritmayı hesaplayıp sonuçları yeni bir diziye yazan versiyon.

```
bool  MathLog(
   const double&   array[],     // değerler dizisi
   const double    base,        // logaritma tabanı
   double&         result[]     // sonuçlar dizisi
  )
```

Belli bir tabana göre logaritmayı hesaplayıp sonuçları girdi dizisinin üstüne yazan versiyon.

```
bool  MathLog(
   double&         array[],     // değerler dizisi
   const double    base         // logaritma tabanı
  )
```

Parametreler

array[]

[in] Değerler dizisi.

base

[in] Logaritma tabanı.

array[]

[out] Sonuç değerlerinin dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathLog2

Dizi elemanlarının 2 tabanında logaritmasını hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathLog2(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathLog2(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathLog10

Dizi elemanlarının 10 tabanında logaritmasını hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathLog10(
   const double&   array[],   // değerler dizisi
   double&         result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathLog10(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathLog1p

Dizi elemanları için log(1+x) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathLog1p(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathLog1p(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathDifference

y[i]=x[i+lag]-x[i] farklarına göre bir dizi oluşturur.

reel değerli dizi ile çalışan versiyon:

```
bool  MathDifference(
   const double  &array[],     // değerler dizisi
   const int     lag,          // gecikme
   double        &result[]     // sonuçlar dizisi
  )
```

Tamsayı değerli dizi ile çalışan versiyon:

```
bool  MathDifference(
   const int     &array[],     // değerler dizisi
   const int     lag,          // gecikme
   int           &result[]     // sonuçlar dizisi
  )
```

Reel değerli bir dizinin tekrarlı oluşumu için kullanılan versiyon (tekrar sayısı giriş parametrelerinde belirtilir):

```
bool  MathDifference(
   const double  &array[],     // değerler dizisi
   const int     lag,          // gecikme
   const int     differences,  // tekrarların sayısı
   double        &result[]     // sonuçlar dizisi
  )
```

Tamsayı değerli bir dizinin tekrarlı oluşumu için kullanılan versiyon (tekrar sayısı giriş parametrelerinde belirtilir):

```
bool  MathDifference(
   const int&    array[],      // değerler dizisi
   const int     lag,          // gecikme
   const int     differences,  // tekrar sayısı
   int&          result[]      // sonuçlar dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

lag

[in] Gecikme parametresi.

differences

[in] Tekrarların sayısı.

result[]

[out] Sonuçların yazılacağı dizi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathSample

Dizi elemanlarından rassal bir örnek oluşturur.

Reel değerli diziler için kullanılan versiyon:

```
bool  MathSample(
   const double&  array[],          // değerler dizisi
   const int      count,            // sayı
   double&        result[]          // sonuçlar dizisi
  )
```

Tamsayı değerli diziler için kullanılan versiyon:

```
bool  MathSample(
   const int&     array[],          // değerler dizisi
   const int      count,            // sayı
   int&           result[]          // sonuçlar dizisi
  )
```

Reel değerli diziler için kullanılan versiyon: İadeli bir örnek elde edilmesi mümkündür:

```
bool  MathSample(
   const double&  array[],          // değerler dizisi
   const int      count,            // sayı
   const bool     replace,          // bayrak
   double&        result[],         // sonuçlar dizisi
  )
```

Tamsayı değerli diziler için kullanılan versiyon: İadeli bir örnek elde edilmesi mümkündür:

```
bool  MathSample(
   const int&     array[],          // değerler dizisi
   const int      count,            // sayı
   const bool     replace,          // bayrak
   int&           result[]          // sonuçlar dizisi
  )
```

Örnekleme olasılıklarının tanımlandığı reel değerli diziler için kullanılan versiyon:

```
bool  MathSample(
   const double&  array[],          // değerler dizisi
   double&        probabilities[],  // olasılıklar dizisi
   const int      count,            // sayı
   double&        result[]          // sonuçlar dizisi
  )
```

Örnekleme olasılıklarının tanımlandığı tamsayı değerli diziler için kullanılan versiyon:

```
bool  MathSample(
```

```
   const int&     array[],           // değerler dizisi
   double&        probabilities[],   // olasılıklar dizisi
   const int      count,             // sayı
   int&           result[]           // sonuçlar dizisi
  )
```

Örnekleme olasılıklarının tanımlandığı reel değerli diziler için kullanılan versiyon: İadeli bir örnek elde edilmesi mümkündür:

```
bool  MathSample(
   const double&  array[],           // değerler dizisi
   double&        probabilities[],   // olasılıklar dizisi
   const int      count,             // sayı
   const bool     replace,           // bayrak
   double&        result[]           // sonuçlar dizisi
  )
```

Örnekleme olasılıklarının tanımlandığı tamsayı değerli diziler için kullanılan versiyon: İadeli bir örnek elde edilmesi mümkündür:

```
bool  MathSample(
   const int&     array[],           // değerler dizisi
   double&        probabilities[],   // olasılıklar dizisi
   const int      count,             // sayı
   const bool     replace,           // bayrak
   int&           result[]           // sonuçlar dizisi
  )
```

Parametreler

array[]

[in] Tamsayılı değerler dizisi.

probabilities[]

[in] Elemanların örneklenmesi için kullanılacak olasılıklar dizisi.

count

[in] Elemanların sayısı.

replace

[in] Örneklemin iadeli olup olmayacağını belirten parametre.

result[]

[out] Sonuçların yazılacağı dizi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Not

replace=true argümanı, örneklemenin iadeli olarak gerçekleştirilmesini sağlar.

MathTukeySummary

Dizi elemanları için Tukey'nin beş-sayılık özetini hesaplar (minimum, alt kuartil, medyan, üst kuartil, maksimum).

```
bool  MathTukeySummary(
   const double&  array[],       // değerler dizisi
   const bool     removeNAN,     // bayrak
   double&        minimum,       // minimum değer
   double&        lower_hinge,   // alt kuartil
   double&        median,        // medyan değeri
   double&        upper_hinge,   // üst kuartil
   double&        maximum        // maksimum değer
  )
```

Parametreler

array[]

[in] Reel değerli dizi.

removeNAN

[in] Nümerik olmayan değerlerin kaldırılıp kaldırılmayacağını belirten bayrak.

minimum

[out] Minimum değerin yazılacağı değişken.

lower_hinge

[out] Alt kuartilin yazılacağı değişken.

median

[out] Medyan değerinin yazılacağı değişken.

upper_hinge

[out] Üst kuartilin yazılacağı değişken.

maximum

[out] Maksimum değerin yazılacağı değişken.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathRange

Dizi elemanlarının maksimum ve minimumlarını hesaplar.

```
bool  MathRange(
   const double&  array[],  // değerler dizisi
   double&        min,      // minimum değer
   double&        max       // maksimum değer
  )
```

Parametreler

array[]

[in] Değerler dizisi.

min

[out] Minimum değerin yazılacağı değişken.

max

[out] Maksimum değerin yazılacağı değişken.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathMin

Dizi elemanlarının minimumuna dönüş yapar.

```
double  MathMin(
   const double&  array[]   // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

Dönüş Değeri

Minimum değer.

MathMax

Dizi elemanlarının maksimumuna dönüş yapar.

```
double  MathMax(
   const double&  array[]   // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

Dönüş Değeri

Maksimum değer.

MathSum

Dizi elemanlarının toplamına dönüş yapar.

```
double  MathSum(
   const double&  array[]   // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

Dönüş Değeri

Elemanların toplamı.

MathProduct

Dizi elemanlarının çarpımına dönüş yapar.

```
double  MathProduct(
   const double&  array[]   // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

Dönüş Değeri

Elemanların çarpımı.

MathStandardDeviation

Dizi elemanlarının standart sapmasını hesaplar.

```
double  MathStandardDeviation(
   const double&  array[]   // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

Dönüş Değeri

Standart sapma.

MathAverageDeviation

Dizi elemanlarının mutlak sapmalarının ortalamasını hesaplar.

```
double  MathAverageDeviation(
   const double&  array[]   // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

Dönüş Değeri

Dizi elemanlarının mutlak sapmalarının ortalaması.

MathMedian

Dizi elemanlarının medyanını hesaplar.

```
double  MathMedian(
   double&  array[]   // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

Dönüş Değeri

Medyan değeri.

MathMean

Dizi elemanlarının ortalamasını hesaplar.

```
double  MathMean(
   const double&  array[]   // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

Dönüş Değeri

Ortalama değer.

MathVariance

Dizi elemanlarının varyansını (ikinci moment) hesaplar.

```
double  MathVariance(
   const double&  array[]   // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

Dönüş Değeri

Varyans değeri.

MathSkewness

Dizi elemanlarının çarpıklığını (üçüncü moment) hesaplar.

```
double  MathSkewness(
   const double&  array[]   // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

Dönüş Değeri

Çarpıklık.

MathKurtosis

Dizi elemanlarının basıklığını (dördüncü moment) hesaplar.

```
double  MathKurtosis(
   const double&  array[]   // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

Dönüş Değeri

Basıklık.

MathExpm1

Dizi elemanları için exp(x)-1 fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathExpm1(
   const double&   array[],   // değerler dizisi
   double&         result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathExpm1(
   double&         array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathSinh

Dizi elemanları için sinh(x) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathSinh(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathSinh(
   double&       array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathCosh

Dizi elemanları için cosh(x) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathCosh(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathCosh(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathTanh

Dizi elemanları için tanh(x) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathTanh(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathTanh(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathArcsinh

Dizi elemanları için arcsinh(x) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathArcsinh(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathArcsinh(
   double&         array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathArccosh

Dizi elemanları için arccosh(x) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathArccosh(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathArccosh(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathArctanh

Dizi elemanları için arctanh(x) fonksiyonunun değerini hesaplar.

Sonuçları yeni bir diziye kaydeden versiyon:

```
bool  MathArctanh(
   const double&  array[],   // değerler dizisi
   double&        result[]   // sonuçlar dizisi
  )
```

Sonuçları girdi dizisinin üstüne kaydeden versiyon:

```
bool  MathArctanh(
   double&        array[]    // değerler dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathSignif

Bir değeri, belirtilen ondalık basamak sayısına yuvarlar.

Reel değerler için kullanılan versiyon:

```
double  MathSignif(
   const double   x,          // value
   const int      digits      // ondalık basamak sayısı
  )
```

Dönüş Değeri

Yuvarlanmış değer.

Reel değerlerle çalışan ve sonuçları yeni bir diziye yazan versiyon.

```
bool  MathSignif(
   const double&  array[],    // değerler dizisi
   int            digits,     // ondalık basamak sayısı
   double         result[]    // sonuçlar dizisi
  )
```

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Reel değerlerle çalışan ve sonuçları orjinal dizinin üstüne yazan versiyon.

```
bool  MathSignif(
   double&        array[],    // değerler dizisi
   int            digits      // ondalık basamak sayısı
  )
```

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Parametreler

x

[in] Yuvarlanacak reel değer.

digits

[in] Ondalık basamak sayısı.

array[]

[in] Reel değerler dizisi.

array[]

[out] Sonuç değerlerinin dizisi.

result[]

[out] Sonuç değerlerinin dizisi.

MathRank

Dizi elemanlarının sırasını hesaplar.

Reel değerli diziler için kullanılan versiyon:

```
bool  MathRank(
   const double&  array[],  // değerler dizisi
   double&        rank[]    // sıralar dizisi
  )
```

Tamsayı değerli diziler için kullanılan versiyon:

```
bool  MathRank(
   const int&     array[],  // değerler dizisi
   double&        rank[]    // sıralar dizisi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

rank[]

[out] Sıraların yazılacağı dizi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathCorrelationPearson

Pearson korelasyon katsayısını hesaplar.

Reel değerli diziler için kullanılan versiyon:

```
bool  MathCorrelationPearson(
   const double&  array1[],  // birinci değerler dizisi
   const double&  array2[],  // ikinci değerler dizisi
   double&        r          // korelasyon katsayısı
  )
```

Tamsayı değerli diziler için kullanılan versiyon:

```
bool  MathCorrelationPearson(
   const int&     array1[],  // birinci değerler dizisi
   const int&     array2[],  // ikinci değerler dizisi
   double&        r          // korelasyon katsayısı
  )
```

Parametreler

array1[]

[in] The first Değerler dizisi.

array2[]

[in] The second Değerler dizisi.

r

[out] Korelasyon katsayısının yazılacağı değişken.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathCorrelationSpearman

Spearman korelasyon katsayısını hesaplar.

Reel değerli diziler için kullanılan versiyon:

```
bool  MathCorrelationSpearman(
   const double&  array1[],  // birinci değerler dizisi
   const double&  array2[],  // ikinci değerler dizisi
   double&        r          // korelasyon katsayısı
  )
```

Tamsayı değerli diziler için kullanılan versiyon:

```
bool  MathCorrelationSpearman(
   const int&     array1[],   // birinci değerler dizisi
   const int&     array2[],   // ikinci değerler dizisi
   double&        r           // korelasyon katsayısı
  )
```

Parametreler

array1[]

[in] The first Değerler dizisi.

array2[]

[in] The second Değerler dizisi.

r

[out] Korelasyon katsayısının yazılacağı değişken.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathCorrelationKendall

Kendall korelasyon katsayısını hesaplar.

Reel değerli diziler için kullanılan versiyon:

```
bool  MathCorrelationKendall(
   const double&  array1[],  // birinci değerler dizisi
   const double&  array2[],  // ikinci değerler dizisi
   double&        tau        // korelasyon katsayısı
  )
```

Tamsayı değerli diziler için kullanılan versiyon:

```
bool  MathCorrelationKendall(
   const int&     array1[],  // birinci değerler dizisi
   const int&     array2[],  // ikinci değerler dizisi
   double&        tau        // korelasyon katsayısı
  )
```

Parametreler

array1[]

[in] The first Değerler dizisi.

array2[]

[in] The second Değerler dizisi.

tau

[out] Korelasyon katsayısının yazılacağı değişken.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathQuantile

belirtilen olasılıklara göre örnek kuantillerini hesaplar: Q[i](p) = (1 - gama)*x[j] + gama*x[j+1]

```
bool  MathQuantile(
   const double&  array[],     // değerler dizisi
   const double&  probs[],     // olasılıklar dizisi
   double&        quantile[]   // kuantillerin yazılacağı dizi
  )
```

Parametreler

array[]

[in] Değerler dizisi.

probs[]

[in] Olasılıklar dizisi.

quantile[]

[out] Kuantillerin yazılacağı dizi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathProbabilityDensityEmpirical

Diziden alınan rassal değerler için ampirik olasılık yoğunluk fonksiyonunu (pdf) hesaplar.

```
bool  MathProbabilityDensityEmpirical(
   const double&  array[],   // rassal değerler dizisi
   const int      count,     // çiftlerin sayısı
   double&        x[],       // x değerlerinin dizisi
   double&        pdf[]      // pdf değerleri dizisi
  )
```

Parametreler

array[]

[in] Rassal değerler dizisi.

count

[in] (x, pdf(x)) çiftlerinin sayısı.

x[]

[out] x değerlerinin yazılacağı dizi.

pdf[]

[out] pdf(x) değerlerinin yazılacağı dizi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

MathCumulativeDistributionEmpirical

Diziden alınan rassal değerler için ampirik birikimli dağılım fonksiyonunu (cdf) hesaplar.

```
bool  MathCumulativeDistributionEmpirical(
   const double&  array[],    // rassal değerler dizisi
   const int      count,      // çiftlerin sayısı
   double&        x[],        // x değerlerinin dizisi
   double&        cdf[]       // cdf değerlerinin dizisi
  )
```

Parametreler

array[]

[in] Rassal değerler dizisi.

count

[in] (x, cdf(x)) çiftlerinin sayısı.

x[]

[out] x değerlerinin yazılacağı dizi.

cdf[]

[out] cdf(x) değerlerinin yazılacağı dizi.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Fuzzy, bulanık mantık ile çalışmak için tasarlanmış bir kütüphanedir

Bulanık mantık, doğru, sözsel bir değişken şeklinde vurgulandığında geleneksel Aristocu mantığın bir sentezidir. Klasik mantıkta olduğu gibi, bulanık mantıkta da konuya özgü -tanımlanan bulanık kümeler üzerinde- mantıksal işlemler bulunur. Bulanık kümeler üzerindeki işlemler sıradan kümelerde olduğu gibidir ama hesaplanmaları daha zordur. Ayrıca belirtmemiz gerekir ki bulanık kümelerin kompozisyonları yine bulanık kümeler oluşturur.

Bulanık mantığı klasik mantıkan ayıran ana unsurlar, gerçeğin yansımasına maksimum derecede yakınlık ve oldukça yüksek seviyede bir öznellik şeklinde tanımlanabilir. Bu durum, hesaplamalarda ciddi hatalara yol açabilmektedir.

Bulanık modeller (veya sistemler), hesaplamaları bulanık mantık temelinde gerçekleştirilen matematiksel modellerdir. Bunlar, çalışma konusunun biçimselliğinin zayıf olması veya matematiksel açıklamasının aşırı karmaşık olması durumunda kurulabilir. Bu modellerin sonuç kalitesi, modeli kuran Uzman Danışmana doğrudan bağlıdır. Hata minimizasyonu için en iyi seçenek, en eksiksiz ve en kapsamlı modelin oluşturulması ve yine bu modelin, geniş eğitim kümeleriyle çalışan makine öğrenimi ile devamlı olarak düzeltilmesiyle elde edilir.

Model oluşturma süreci üç ana aşamaya bölünebilir:

1. Modelin girdi ve çıktı özelliklerinin tanımlanması.
2. Bilgi temelinin kurulması.
3. Bulanık çıkarım yöntemlerinden (Mamdani veya Sugeno) birinin seçimi.

İlk aşama, sonraki iki aşamayı da etkiler ve modelin gelecekteki işlemleri için belirleyicidir. Bilgi tabanı (veya diğer ismiyle kural tabanı), üstünde çalışılan nesnenin girdileri ve çıktıları arasındaki bağıntıyı "eğer" ve "ise" gibi işlemlerle tanımlayan bir bulanık kurallar kümesidir. Sistemdeki kuralların sayısı Uzman Danışman tarafından belirlenir ve sınırsızdır. Bulanık kuralların genel çerçevesi şu şekildedir:

Kural durumu ise, kural sonucu.

Kural durumu nesnenin mevcut durumunu, kural sonucu ise durumun nesneyi nasıl etkilediğini açıklar. Durumların ve sonuçların genel durumu kullanıcı tarafından seçilemez; bulanık çıkarım ile belirlenirler.

Sistemdeki her kuralın bir ağırlığı vardır, bu ağırlık kuralın modeldeki önemini gösterir. Kurallara atanan ağırlıklar [0, 1] aralığında olmalıdır. Bulanık modellerle ilgi literatürde rastlanan bir çok örnekte ağırlık verileri belirtilmemiştir, ama bu ağırlığın olmadığı anlamına gelmez. Aslında bu tip durumlarda veritabanındaki her bir kural için ağırlık sabittir ve bire eşittir. Her Kural için iki tip terim ve sonuç olabilir:

1. basit — bir bulanık değişken içerir;
2. karmaşık — birkaç bulanık değişken içerir.

Oluşturulan bilgi temeline göre, model için kullanılacak bulanık çıkarım sistemi belirlenir. Bulanık mantıksal çıkarım, bilgi tabanı ve bulanık işlemlerin kullanılmasının ardından, sonuçların, girdilerin mevcut durumunu gösteren bir bulanık küme şeklinde elde edilmesidir. Bulanık çıkarımın "Mamdani" ve "Sugeno" olmak üzere iki tipi bulunur.

Üyelik fonksiyonları

Üyelik fonksiyonları evrensel uzaydaki rassal bir elemanın belirtilen bulanık kümeye ne derecede üye olduğunun hesaplanmasını sağlar. [0, 1] kapalı aralığında tanımlıdırlar.

Çoğu durumda, sürekli ve monotondurlar.

Üyelik fonksiyonlarının sınıfları	Açıklama
CConstantMembershipFunction	Üyelik fonksiyonunu koordinat eksenine paralel, düz bir çizgi şeklinde uygulamak için tasarlanmıştır
CCompositeMembershipFunction	Üyelik fonksiyonlarının bir kompozisyonu ile çalışmak için tasarlanmıştır
CDifferencTwoSigmoidalMembershipFunction	Üyelik fonksiyonunu; A1, A2, C1 ve C2 parametrelerine sahip iki sigmoid fonksiyonun farkı şeklinde kullanmak için tasarlanmıştır
CGeneralizedBellShapedMembershipFunction	Çan eğrisi şeklinde ve A, B, C parametrelerine sahip üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır
CNormalCombinationMembershipFunction	B1, B2, Sigma1 ve Sigma2 parametrelerine sahip çift taraflı Gaussyen üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır
CNormalMembershipFunction	B ve Sigma parametrelerine sahip simetrik Gaussyen üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır
CP_ShapedMembershipFunction	A, B, C, D parametrelerine sahip Pi-şekilli üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır
CProductTwoSigmoidalMembershipFunction	Üyelik fonksiyonunu; A1, A2, C1 ve C2 parametrelerine sahip iki sigmoid fonksiyonun çarpımı şeklinde kullanmak için tasarlanmıştır
CS_ShapedMembershipFunction	A ve B parametrelerine sahip S-şekilli üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır
CTrapezoidMembershipFunction	X1, X2, X3 ve X4 parametrelerine sahip trapezoit üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır
CTriangularMembershipFunction	X1, X2 ve X3 parametrelerine sahip üçgen üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır
CSigmoidalMembershipFunction	A ve C parametrelerine sahip sigmoid üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır

Üyelik fonksiyonlarının sınıfları	Açıklama
CZ_ShapedMembershipFunction	A ve B parametrelerine sahip z-şekilli üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır
IMembershipFunction	Tüm üyelik fonksiyonu sınıfları için temel sınıf.

CConstantMembershipFunction

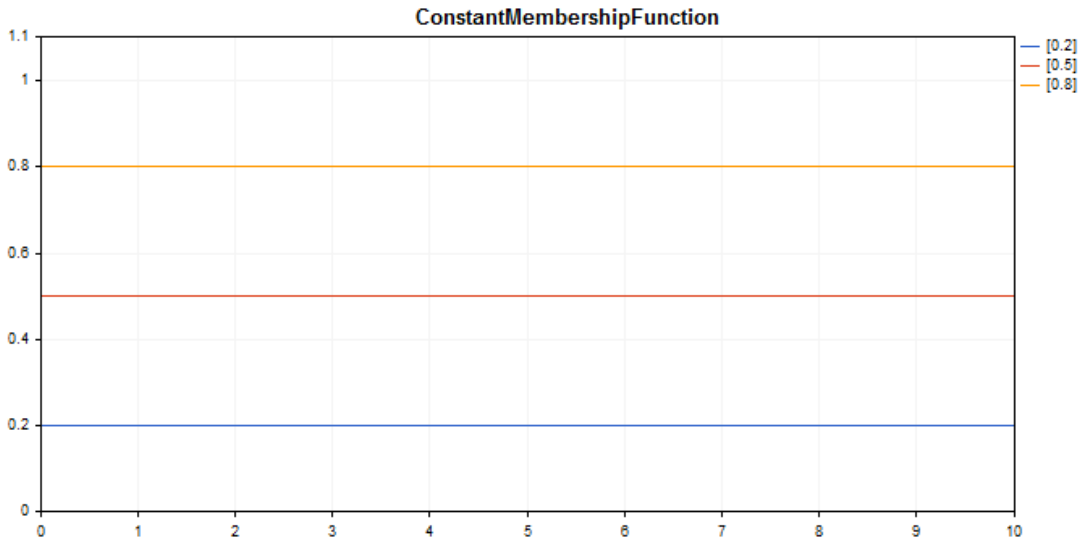
Üyelik fonksiyonunu koordinat eksenine paralel, düz bir çizgi şeklinde uygulamak için tasarlanmıştır.

Açıklama

Fonksiyon şu eşitlikle tanımlanır:

y(x)=c

Yani, fonksiyonun üyelik derecesi tüm sayısal eksen üzerinde aynıdır ve yapıcıda belirtilen parametreye eşittir.



Çizelge çizmek için oluşturulmuş bir örnek kod aşağıda verilmiştir.

Bildirim

```
class CConstantMembershipFuncion : public IMembershipFunction
```

Başlık

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Kalıtım hiyerarşisi

CObject
IMembershipFunction
CConstantMembershipFunction

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
GetValue	Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Örnek

```
//+------------------------------------------------------------------+
//|                                  ConstantMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Üyelik fonksiyonlarını oluştur
CConstantMembershipFunction func1(0.2);
CConstantMembershipFunction func2(0.5);
CConstantMembershipFunction func3(0.8);
//--- üyelik fonksiyonları için örtüler oluştur
double ConstantMembershipFunction1(double x) { return(func1.GetValue(x)); }
double ConstantMembershipFunction2(double x) { return(func2.GetValue(x)); }
double ConstantMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- grafiği oluştur
   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);
//--- eğri oluştur
   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]");
//--- X-ekseninin özelliklerini ayarla
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y-ekseninin özelliklerini ayarla
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- çiz
```

```
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

GetValue

Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

```
double  GetValue(
   const double  x      // üyelik fonksiyonunun argümanı
   )
```

Parametreler

x

[in] Üyelik fonksiyonunun argümanı.

Dönüş Değeri

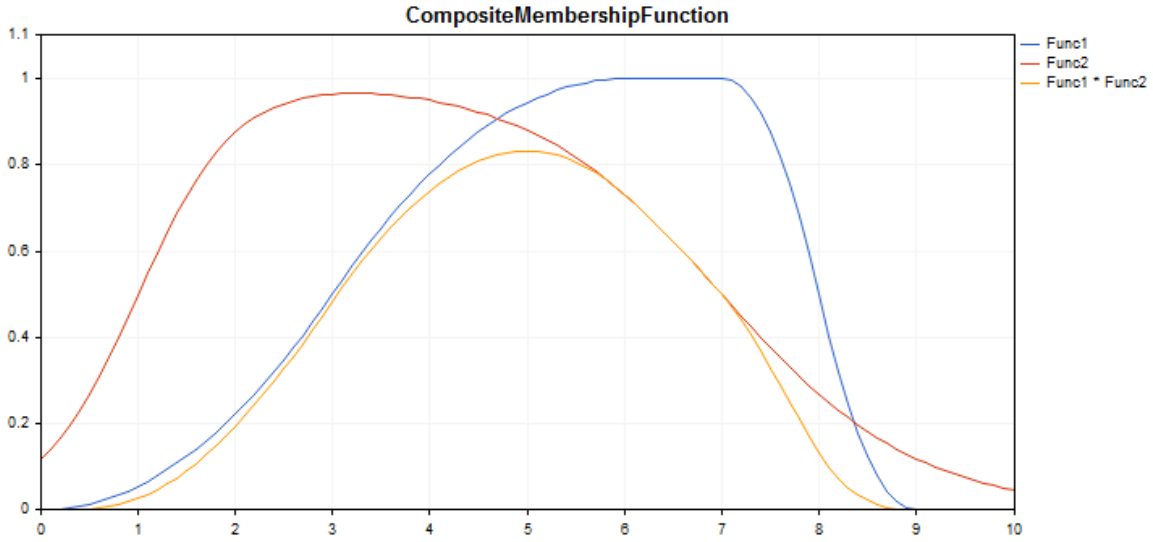
Üyelik fonksiyonunun değeri

CCompositeMembershipFunction

Üyelik fonksiyonlarının bir kompozisyonunun uygulanması için tasarlanan bir sınıftır.

Açıklama

Üyelik fonksiyonlarının kompozisyonu, belirtilen operatörü içeren iki veya daha fazla üyelik fonksiyonunun bir kombinasyonudur.



Çizelge çizmek için oluşturulmuş bir örnek kod aşağıda verilmiştir.

Bildirim

```
class CCompositeMembershipFuncion : public IMembershipFunction
```

Başlık

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Kalıtım hiyerarşisi

CObject
  IMembershipFunction
    CCompositeMembershipFunction

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
CompositionType	Kompozisyon operatörünü ayarlar.
MembershipFunctions	Üyelik fonksiyonlarının listesini alır.
GetValue	Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Örnek

```
//+------------------------------------------------------------------+
//|                                  CompositeMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Üyelik fonksiyonlarını oluştur
CProductTwoSigmoidalMembershipFunctions func1(2,1,-1,7);
CP_ShapedMembershipFunction func2(0,6,7,9);
uCCompositeMembershipFunction composite(ProdMF,GetPointer(func1),GetPointer(func2));
//--- üyelik fonksiyonları için örtüler oluştur
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)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- grafiği oluştur
   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);
//--- eğri oluştur
   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
//--- X-ekseninin özelliklerini ayarla
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y-ekseninin özelliklerini ayarla
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- çiz
```

```
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

CompositionType

Kompozisyon operatörünü ayarlar.

```
void  CompositionType(
   MfCompositionType  value      // operatör tipi
   )
```

Parametreler

value

[in] Kompozisyon operatörünün tipi.

Not

Şu operatör tipleri mevcuttur:

- MinMF (fonksiyonların minimumu)
- MaxMF (fonksiyonların maksimumu)
- ProdMF (fonksiyonların çarpımı)
- SumMF (fonksiyonların toplamı)

MembershipFunctions

Bir kompozisyona eklenmiş üyelik fonksiyonlarının listesini alır.

```
CList*  MembershipFunctions(
   void        // üyelik fonksiyonlarının listesi
   )
```

Dönüş Değeri

Üyelik fonksiyonlarının listesi.

GetValue

Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

```
double  GetValue(
   const  x      // üyelik fonksiyonunun argümanı
   )
```

Parametreler

x

[in] Üyelik fonksiyonunun argümanı.

Dönüş Değeri

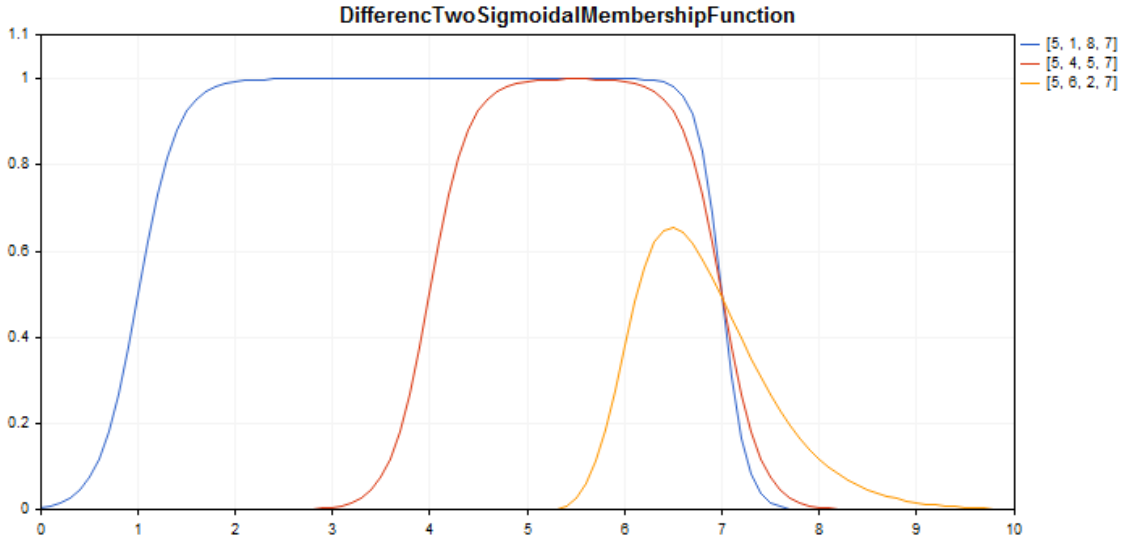
Üyelik fonksiyonunun değeri

CDifferencTwoSigmoidalMembershipFunction

Üyelik fonksiyonunu; A1, A2, C1 ve C2 parametrelerine sahip iki sigmoid fonksiyonun farkı şeklinde kullanmak için tasarlanmıştır.

Açıklama

Fonksiyon bir sigmoid eğri temelinde ayarlanır. Argüman değeriyle başlayan ve değeri bire eşit olan üyelik fonksiyonlarının oluşturulmasını sağlar. Bu tip fonksiyonlar "kısa" veya "uzun" gibi sözel terimlerle çalışmak istediğinizde kullanılabilir.



Çizelge çizmek için oluşturulmuş bir örnek kod aşağıda verilmiştir.

Bildirim

```
class CDifferencTwoSigmoidalMembershipFuncion : public IMembershipFunction
```

Başlık

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Kalıtım hiyerarşisi

CObject

IMembershipFunction

CDifferencTwoSigmoidalMembershipFunction

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
A1	İlk üyelik fonksiyonunun eğim oranını alır/ayarlar.
A2	İkinci üyelik fonksiyonunun eğim oranını alır/ayarlar.

Sınıf yöntemi	Açıklama
C1	İlk üyelik fonksiyonunun dönüm koordinatı parametresini alır/ayarlar.
C2	İkinci üyelik fonksiyonunun dönüm koordinatı parametresini alır/ayarlar.
GetValue	Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Örnek

```
//+------------------------------------------------------------------+
//|                      DifferencTwoSigmoidalMembershipFunction.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Üyelik fonksiyonlarını oluştur
CDifferencTwoSigmoidalMembershipFunction func1(5,1,8,7);
CDifferencTwoSigmoidalMembershipFunction func2(5,4,5,7);
CDifferencTwoSigmoidalMembershipFunction func3(5,6,2,7);
//--- üyelik fonksiyonları için örtüler oluştur
double DifferencTwoSigmoidalMembershipFunction1(double x) { return(func1.GetValue(x));
double DifferencTwoSigmoidalMembershipFunction2(double x) { return(func2.GetValue(x));
double DifferencTwoSigmoidalMembershipFunction3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- grafiği oluştur
   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);
//--- eğri oluştur
   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,
//--- X-ekseninin özelliklerini ayarla
```

```
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y-ekseninin özelliklerini ayarla
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- çiz
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A1 (Get yöntemi)

İlk üyelik fonksiyonunun eğim oranını alır.

```
double  A1()
```

Dönüş Değeri

Eğim oranı değeri.

A1 (Set yöntemi)

İlk üyelik fonksiyonunun eğim oranını ayarlar.

```
void  A1(
   const double  a1      // eğim oranı değeri
   )
```

Parametreler

a1

[in] Eğim oranı değeri.

A2 (Get yöntemi)

İkinci üyelik fonksiyonunun eğim oranını alır.

```
double  A2()
```

Dönüş Değeri

Eğim oranı değeri.

A2 (Set yöntemi)

İkinci üyelik fonksiyonunun eğim oranını ayarlar.

```
void  A2(
   const double  a2      // eğim oranı değeri
```

```
   )
```

Parametreler

a2

[in] Eğim oranı değeri.

C1 (Get yöntemi)

İlk üyelik fonksiyonunun dönüm koordinatı parametresini alır.

```
double  C1()
```

Dönüş Değeri

Bükülme koordinatı değeri.

C1 (Set yöntemi)

İlk üyelik fonksiyonunun dönüm koordinatı parametresini ayarlar.

```
void  C1(
   const double  c1      // bükülme koordinatı değeri
   )
```

Parametreler

c1

[in] Bükülme koordinatı değeri.

C2 (Get yöntemi)

İkinci üyelik fonksiyonunun dönüm koordinatı parametresini alır.

```
double  C2()
```

Dönüş Değeri

Bükülme koordinatı değeri.

C2 (Set yöntemi)

İkinci üyelik fonksiyonunun dönüm koordinatı parametresini ayarlar.

```
void  C2(
   const double  c2      // bükülme koordinatı değeri
   )
```

Parametreler

c2

[in] Bükülme koordinatı değeri.

GetValue

Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

```
double  GetValue(
   const double  x      // üyelik fonksiyonunun argümanı
   )
```

Parametreler

x

[in] Üyelik fonksiyonunun argümanı.

Dönüş Değeri

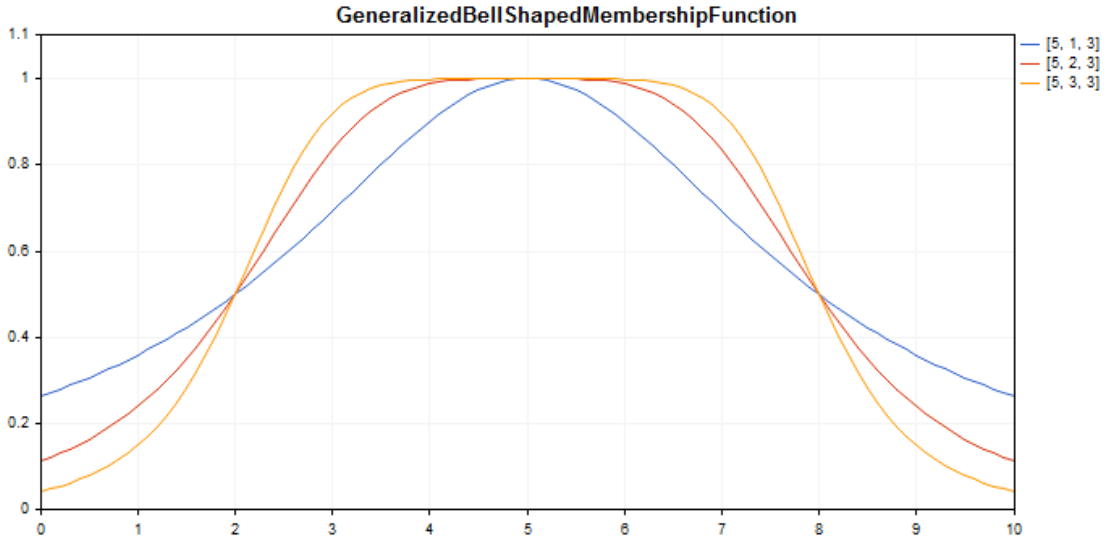
Üyelik fonksiyonunun değeri

CGeneralizedBellShapedMembershipFunction

Çan eğrisi şeklinde ve A, B, C parametrelerine sahip üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır

Açıklama

Genel çan eğrisi şekilli fonksiyonlar Gaussyen fonksiyonlara benzer. Fonksiyon düzgündür ve tüm tanım aralığı boyunca sıfırdan farklı değerler alır.



Çizelge çizmek için oluşturulmuş bir örnek kod aşağıda verilmiştir.

Bildirim

```
class CGeneralizedBellShapedMembershipFuncion : public IMembershipFunction
```

Başlık

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Kalıtım hiyerarşisi

CObject
IMembershipFunction
CGeneralizedBellShapedMembershipFunction

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
A	Üyelik fonksiyonunun konsantrasyon oranını alır/ayarlar.
B	Üyelik fonksiyonunun eğim oranını alır/ayarlar.

Sınıf yöntemi	Açıklama
C	Üyelik fonksiyonunun maksimum koordinatını alır/ayarlar.
GetValue	Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Örnek

```
//+------------------------------------------------------------------+
//|                      GeneralizedBellShapedMembershipFunction.mq5 |
//|                         Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Üyelik fonksiyonlarını oluştur
CGeneralizedBellShapedMembershipFunction func1(5, 1, 3);
CGeneralizedBellShapedMembershipFunction func2(5, 2, 3);
CGeneralizedBellShapedMembershipFunction func3(5, 3, 3);
//--- üyelik fonksiyonları için örtüler oluştur
double GeneralizedBellShapedMembershipFunction1(double x) { return(func1.GetValue(x));
double GeneralizedBellShapedMembershipFunction2(double x) { return(func2.GetValue(x));
double GeneralizedBellShapedMembershipFunction3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- grafiği oluştur
   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);
//--- eğri oluştur
   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,
//--- X-ekseninin özelliklerini ayarla
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y-ekseninin özelliklerini ayarla
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- çiz
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Get yöntemi)

Üyelik fonksiyonunun konsantrasyon oranını alır.

```
double  A()
```

Dönüş Değeri

Konsantrasyon oranının değeri.

A (Set yöntemi)

Üyelik fonksiyonunun konsantrasyon oranını ayarlar.

```
void  A(
   const double  a      // konsantrasyon oranının değeri
   )
```

Parametreler

a

[in] Üyelik fonksiyonunun konsantrasyon oranı.

B (Get yöntemi)

Üyelik fonksiyonunun eğim oranını alır.

```
double  B()
```

Dönüş Değeri

Eğim oranı değeri.

B (Set yöntemi)

Üyelik fonksiyonunun eğim oranını ayarlar.

```
void  B(
   const double  b      // eğim oranı değeri
   )
```

Parametreler

b

[in] Üyelik fonksiyonu eğim oranı.

C (Get yöntemi)

Üyelik fonksiyonunun maksimum koordinatını alır.

```
double  C()
```

Dönüş Değeri

Üyelik fonksiyonunun maksimum koordinatı.

C (Set yöntemi)

Üyelik fonksiyonunun maksimum koordinatını ayarlar.

```
void  C(
   const double  c      // maksimum koordinat değeri
   )
```

Parametreler

c

[in] Üyelik fonksiyonunun maksimum koordinatı.

GetValue

Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

```
double  GetValue(
   const  x      // üyelik fonksiyonunun argümanı
   )
```

Parametreler

x

[in] Üyelik fonksiyonunun argümanı.

Dönüş Değeri

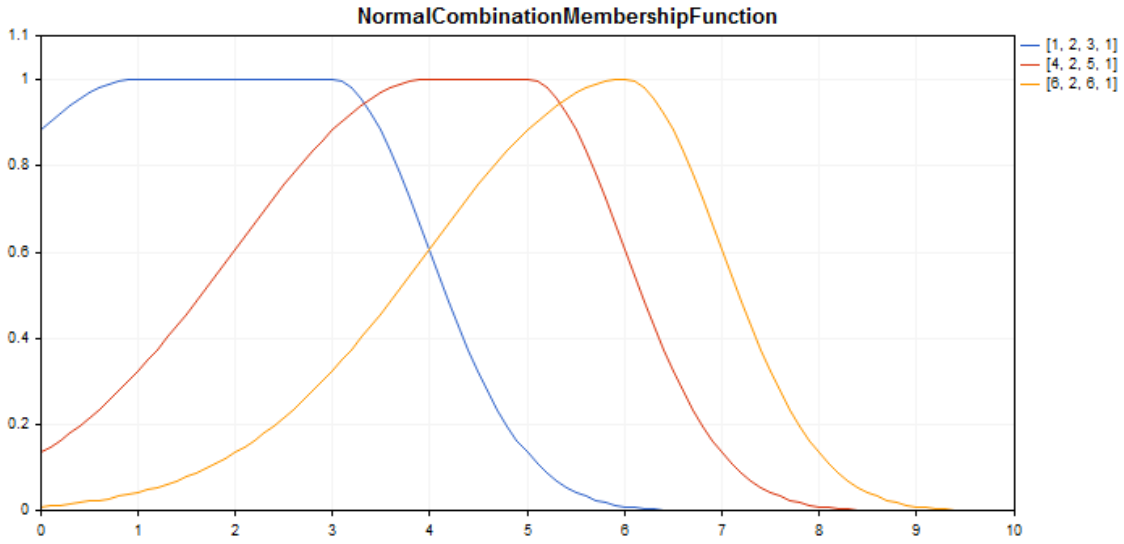
Üyelik fonksiyonunun değeri

CNormalCombinationMembershipFunction

B1, B2, Sigma1 ve Sigma2 parametrelerine sahip çift taraflı Gaussyen üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır

Açıklama

Çift taraflı Gaussyen üyelik fonksiyonu Gaussyen (normal) dağılım ile şekillendirilir Asimetrik üyelik fonksiyonlarının kullanılabilmesini sağlar. Fonksiyon düzgündür ve tüm tanım aralığı boyunca sıfırdan farklı değerler alır.



Çizelge çizmek için oluşturulmuş bir örnek kod aşağıda verilmiştir.

Bildirim

```
class CNormalCombinationMembershipFuncion : public IMembershipFunction
```

Başlık

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Kalıtım hiyerarşisi

CObject

IMembershipFunction

CNormalCombinationMembershipFunction

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
B1	İlk üyelik fonksiyonu merkezinin değerini alır/ayarlar.

Sınıf yöntemi	Açıklama
B2	İkinci üyelik fonksiyonu merkezinin değerini alır/ayarlar.
Sigma1	Üyelik fonksiyonunun ilk eğrilik parametresini alır/ayarlar.
Sigma2	Üyelik fonksiyonunun ikinci eğrilik parametresini alır/ayarlar.
GetValue	Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Örnek

```
//+------------------------------------------------------------------+
//|                          NormalCombinationMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Üyelik fonksiyonlarını oluştur
CNormalCombinationMembershipFunction func1(1,2,3,1);
CNormalCombinationMembershipFunction func2(4,2,5,1);
CNormalCombinationMembershipFunction func3(6,2,6,1);
//--- üyelik fonksiyonları için örtüler oluştur
double NormalCombinationMembershipFunction1(double x) { return(func1.GetValue(x)); }
double NormalCombinationMembershipFunction2(double x) { return(func2.GetValue(x)); }
double NormalCombinationMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- grafiği oluştur
   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);
//--- eğri oluştur
   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,
//--- X-ekseninin özelliklerini ayarla
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y-ekseninin özelliklerini ayarla
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- çiz
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

B1 (Get yöntemi)

İlk üyelik fonksiyonu merkezinin değerini alır.

```
double  B1()
```

Dönüş Değeri

İlk üyelik fonksiyonu merkezinin değeri.

B1 (Set yöntemi)

İlk üyelik fonksiyonu merkezinin değerini ayarlar.

```
void  B1(
   const double  b1      // ilk merkez değeri
   )
```

Parametreler

b

[in] İlk üyelik fonksiyonu merkezinin değeri.

B2 (Get yöntemi)

İkinci üyelik fonksiyonu merkezinin değerini alır.

```
double  B2()
```

Dönüş Değeri

İkinci üyelik fonksiyonu merkezinin değeri.

B2 (Set yöntemi)

İkinci üyelik fonksiyonu merkezinin değerini ayarlar.

```
void  B2(
   const double  b2      // ikinci merkez değeri
   )
```

Parametreler

b2

[in] İkinci üyelik fonksiyonu merkezinin değeri.

Sigma1 (Get yöntemi)

Üyelik fonksiyonunun ilk eğrilik parametresini alır.

```
double  Sigma1()
```

Dönüş Değeri

Üyelik fonksiyonunun ilk eğrilik parametresinin değeri.

Sigma1 (Set yöntemi)

Üyelik fonksiyonunun ilk eğrilik parametresini ayarlar.

```
void  Sigma1(
   const double  sigma1      // ilk eğrilik parametresinin değeri
   )
```

Parametreler

sigma1

[in] Üyelik fonksiyonunun ilk eğrilik parametresi.

Sigma2 (Get yöntemi)

Üyelik fonksiyonunun ikinci eğrilik parametresini alır.

```
double  Sigma2()
```

Dönüş Değeri

Üyelik fonksiyonunun ikinci eğrilik parametresinin değeri.

Sigma2 (Set yöntemi)

Üyelik fonksiyonunun ikinci eğrilik parametresini ayarlar.

```
void  Sigma2(
   const double  sigma2      // ikinci eğrilik parametresinin değeri
   )
```

Parametreler

sigma2

[in] Üyelik fonksiyonunun ikinci eğrilik parametresi.

GetValue

Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

```
double  GetValue(
   const  x      // üyelik fonksiyonunun argümanı
   )
```

Parametreler

x

[in] Üyelik fonksiyonunun argümanı.

Dönüş Değeri

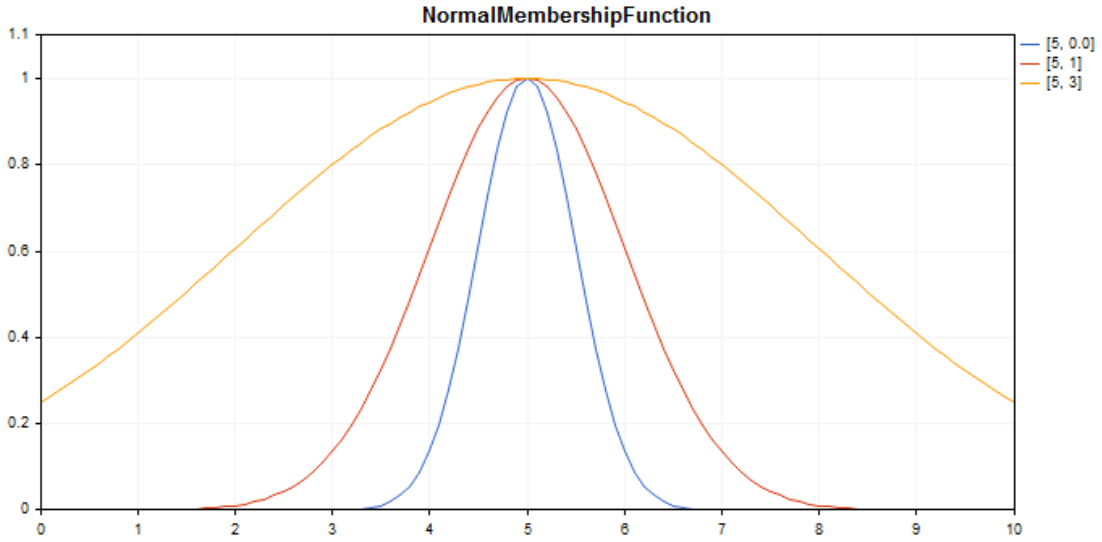
Üyelik fonksiyonunun değeri

CNormalMembershipFunction

B ve Sigma parametrelerine sahip simetrik Gaussyen üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır

Açıklama

Simetrik Gaussyen üyelik fonksiyonu Gaussyen (normal) dağılım ile şekillendirilir Fonksiyon düzgündür ve tüm tanım aralığı boyunca sıfırdan farklı değerler alır.



Çizelge çizmek için oluşturulmuş bir örnek kod aşağıda verilmiştir.

Bildirim

```
class CNormalMembershipFuncion : public IMembershipFunction
```

Başlık

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Kalıtım hiyerarşisi

CObject
IMembershipFunction
CNormalMembershipFunction

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
B	Üyelik fonksiyonu merkezinin değerini alır/ayarlar.
Sigma	Üyelik fonksiyonunun eğrilik parametresini alır/ayarlar.

Sınıf yöntemi	Açıklama
GetValue	Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Örnek

```
//+------------------------------------------------------------------+
//|                                    NormalMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Üyelik fonksiyonlarını oluştur
CNormalMembershipFunction func1(5,0.5);
CNormalMembershipFunction func2(5,1);
CNormalMembershipFunction func3(5,3);
//--- üyelik fonksiyonları için örtüler oluştur
double NormalMembershipFunction1(double x) { return(func1.GetValue(x)); }
double NormalMembershipFunction2(double x) { return(func2.GetValue(x)); }
double NormalMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- grafiği oluştur
   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);
//--- eğri oluştur
   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]");
//--- X-ekseninin özelliklerini ayarla
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y-ekseninin özelliklerini ayarla
```

```
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- çiz
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

B (Get yöntemi)

Üyelik fonksiyonu merkezinin değerini alır.

```
double  B()
```

Dönüş Değeri

Üyelik fonksiyonu merkezinin değeri.

B (Set yöntemi)

Üyelik fonksiyonu merkezinin değerini ayarlar.

```
void  B(
   const double  b      // üyelik fonksiyonu merkezinin değeri
   )
```

Parametreler

b

[in] Üyelik fonksiyonu merkezinin değeri.

Sigma (Get yöntemi)

Üyelik fonksiyonunun eğrilik parametresini alır.

```
double  Sigma()
```

Dönüş Değeri

Üyelik fonksiyonunun eğrilik parametresinin değeri.

Sigma (Set yöntemi)

Üyelik fonksiyonunun eğrilik parametresinin değerini ayarlar.

```
void  Sigma(
   const double  sigma      // eğrilik parametresinin değeri
   )
```

Parametreler

sigma

[in] Üyelik fonksiyonunun eğrilik parametresi.

GetValue

Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

```
double  GetValue(
   const double  x      // argüman
   )
```

Parametreler

x

[in] Üyelik fonksiyonunun argümanı.

Dönüş Değeri

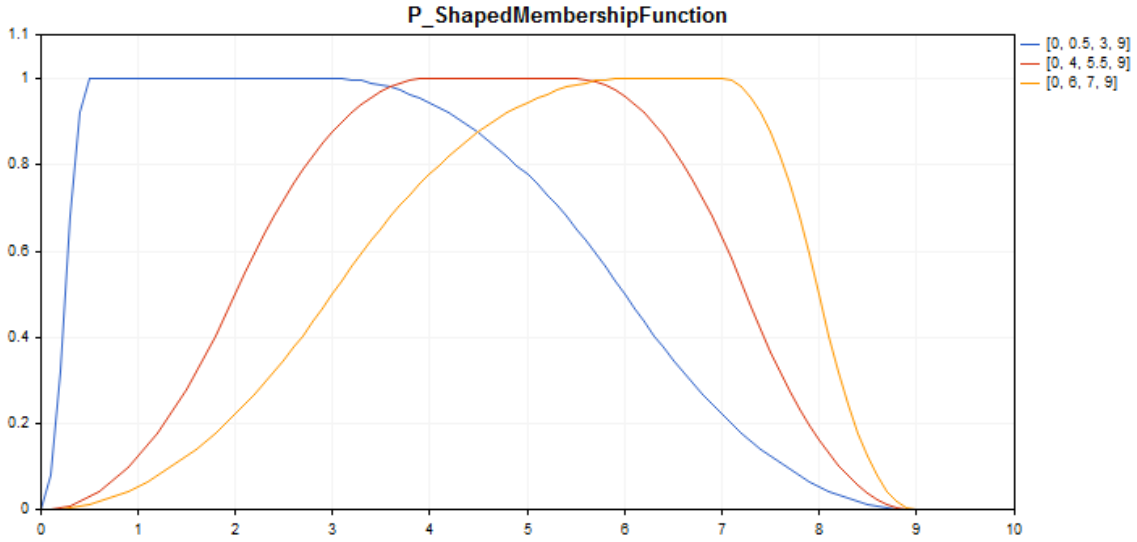
Üyelik fonksiyonunun değeri

CP_ShapedMembershipFunction

A, B, C, D parametrelerine sahip Pi-şekilli üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır

Açıklama

Pi-şekilli üyelik fonksiyonları eğrisel trapezoit şeklindedir. Fonksiyon, değişkenlerin -karamsar bulanık değişken değerlendirmesinden iyimsere düzgün geçişli- asimetrik üyelik fonksiyonlarını ayarlamak için kullanılır.



Çizelge çizmek için oluşturulmuş bir örnek kod aşağıda verilmiştir.

Bildirim

```
class CP_ShapedMembershipFuncion : public IMembershipFunction
```

Başlık

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Kalıtım hiyerarşisi

CObject

IMembershipFunction

CP_ShapedMembershipFunction

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
A	Bulanık küme başlangıcının değerini gösteren parametreyi alır/ayarlar.
B	Bulanık küme çekirdeğinin ilk parametresini alır/ayarlar.

Sınıf yöntemi	Açıklama
C	Bulanık küme çekirdeğinin ikinci parametresini alır/ayarlar.
D	Bulanık kümenin bitiş değerini gösteren parametreyi alır/ayarlar.
GetValue	Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Örnek

```
//+------------------------------------------------------------------+
//|                                   P_ShapedMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Üyelik fonksiyonlarını oluştur
CP_ShapedMembershipFunction func1(0,0.5,3,9);
CP_ShapedMembershipFunction func2(0,4,5.5,9);
CP_ShapedMembershipFunction func3(0,6,7,9);
//--- üyelik fonksiyonları için örtüler oluştur
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)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- grafiği oluştur
   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);
//--- eğri oluştur
   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
//--- X-ekseninin özelliklerini ayarla
```

```
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y-ekseninin özelliklerini ayarla
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- çiz
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Get yöntemi)

Bulanık kümenin başlangıç değerini gösteren parametreyi alır.

```
double  A()
```

Dönüş Değeri

Bulanık kümenin başlangıcının parametresi.

A (Set yöntemi)

Bulanık kümenin başlangıç değerini gösteren parametreyi ayarlar.

```
void  A(
   const double  a      // Bulanık kümenin başlangıç parametresi
   )
```

Parametreler

a

[in] Bulanık kümenin başlangıcının parametresi.

B (Get yöntemi)

Bulanık küme çekirdeğinin ilk parametresini alır.

```
double  B()
```

Dönüş Değeri

Bulanık küme çekirdeğinin ilk parametresi.

B (Set yöntemi)

Bulanık küme çekirdeğinin ilk parametresini ayarlar.

```
void  B(
   const double  b      // bulanık küme çekirdeğinin ilk parametresinin değeri
```

```
   )
```

Parametreler

b

[in] Bulanık küme çekirdeğinin ilk parametresi.

C (Get yöntemi)

Bulanık küme çekirdeğinin ikinci parametresini alır.

```
double  C()
```

Dönüş Değeri

Bulanık küme çekirdeğinin ikinci parametresi.

C (Set yöntemi)

Bulanık küme çekirdeğinin ikinci parametresini ayarlar.

```
void  C(
   const double  c      // bulanık küme çekirdeğinin ikinci parametresinin değeri
   )
```

Parametreler

c

[in] Bulanık küme çekirdeğinin ikinci parametresi.

D (Get yöntemi)

Bulanık kümenin bitiş parametresini alır.

```
double  D()
```

Dönüş Değeri

Bulanık kümenin bitiş parametresinin değeri.

D (Set yöntemi)

Bulanık kümenin bitiş parametresini ayarlar.

```
void  D(
   const double  d      // bulanık kümenin bitiş parametresinin değeri
   )
```

Parametreler

d

[in] Bulanık kümenin bitiş parametresinin değeri.

GetValue

Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

```
double  GetValue(
   const double  x
   )
```

Parametreler

x

[in] Üyelik fonksiyonunun argümanı

Dönüş Değeri

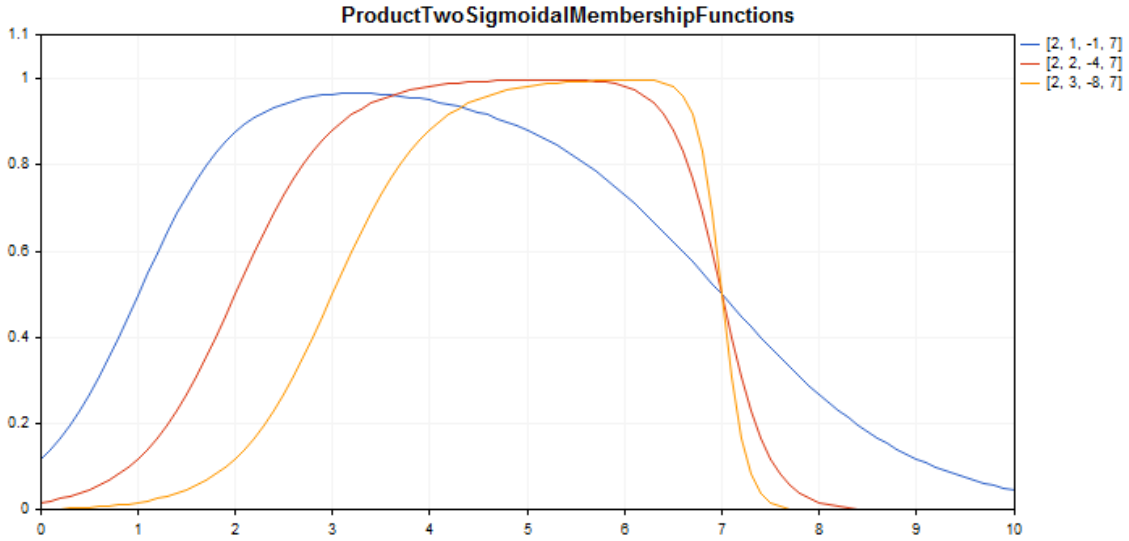
Üyelik fonksiyonunun değeri

CProductTwoSigmoidalMembershipFunction

Üyelik fonksiyonunu; A1, A2, C1 ve C2 parametrelerine sahip iki sigmoid fonksiyonun çarpımı şeklinde kullanmak için tasarlanmıştır.

Açıklama

İki sigmoid üyelik fonksiyonun çarpımı, düzgün asimetrik fonksiyonlarla çalışmak için kullanılır Argüman değeriyle başlayan ve değeri bire eşit olan üyelik fonksiyonlarının oluşturulmasını sağlar. Bu tip fonksiyonlar "kısa" veya "uzun" gibi sözel terimlerle çalışmak istediğinizde kullanılabilir.



Çizelge çizmek için oluşturulmuş bir örnek kod aşağıda verilmiştir.

Bildirim

```
class CProductTwoSigmoidalMembershipFuncion : public IMembershipFunction
```

Başlık

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Kalıtım hiyerarşisi

CObject

IMembershipFunction

CProductTwoSigmoidalMembershipFunctions

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
A1	İlk üyelik fonksiyonunun eğim oranını alır/ayarlar.
A2	İkinci üyelik fonksiyonunun eğim oranını alır/ayarlar.

Sınıf yöntemi	Açıklama
C1	İlk üyelik fonksiyonunun dönüm koordinatı parametresini alır.
C2	İkinci üyelik fonksiyonunun dönüm koordinatı parametresini alır.
GetValue	Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Örnek

```
//+------------------------------------------------------------------+
//|                      ProductTwoSigmoidalMembershipFunctions.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             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>
//--- Üyelik fonksiyonlarını oluştur
CProductTwoSigmoidalMembershipFunctions func1(2,1,-1,7);
CProductTwoSigmoidalMembershipFunctions func2(2,2,-4,7);
CProductTwoSigmoidalMembershipFunctions func3(2,3,-8,7);
//--- üyelik fonksiyonları için örtüler oluştur
double ProductTwoSigmoidalMembershipFunctions1(double x) { return(func1.GetValue(x));
double ProductTwoSigmoidalMembershipFunctions2(double x) { return(func2.GetValue(x));
double ProductTwoSigmoidalMembershipFunctions3(double x) { return(func3.GetValue(x));
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- grafiği oluştur
   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);
//--- eğri oluştur
   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,"
//--- X-ekseninin özelliklerini ayarla
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y-ekseninin özelliklerini ayarla
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- çiz
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A1 (Get yöntemi)

İlk üyelik fonksiyonunun eğim oranını alır.

```
double  A1()
```

Dönüş Değeri

İlk üyelik fonksiyonu eğim oranı.

A1 (Set yöntemi)

İlk üyelik fonksiyonunun eğim oranını ayarlar.

```
void  A1(
   const double  a1      // ilk üyelik fonksiyonu eğim oranı
   )
```

Parametreler

a1

[in] İlk üyelik fonksiyonu eğim oranı.

A2 (Get yöntemi)

İkinci üyelik fonksiyonunun eğim oranını alır.

```
double  A2()
```

Dönüş Değeri

İkinci üyelik fonksiyonu eğim oranı.

A2 (Set yöntemi)

İkinci üyelik fonksiyonunun eğim oranını ayarlar.

```
void  A2(
   const double  a2      // ikinci üyelik fonksiyonunun eğim oranı
   )
```

Parametreler

a2

[in] İkinci üyelik fonksiyonu eğim oranı.

C1 (Get yöntemi)

İlk üyelik fonksiyonunun dönüm koordinatı parametresini alır.

```
double  C1()
```

Dönüş Değeri

İlk üyelik fonksiyonunun dönüm koordinatı.

C1 (Set yöntemi)

İlk üyelik fonksiyonunun dönüm koordinatını ayarlar.

```
void  C1(
   const double  c1      // ilk üyelik fonksiyonunun dönüm koordinatı
   )
```

Parametreler

c1

[in] İlk üyelik fonksiyonunun dönüm koordinatı.

C2 (Get yöntemi)

İkinci üyelik fonksiyonunun dönüm koordinatı parametresini alır.

```
double  C2()
```

Dönüş Değeri

İkinci üyelik fonksiyonunun dönüm koordinatı.

C2 (Set yöntemi)

İkinci üyelik fonksiyonunun dönüm koordinatını ayarlar.

```
void  C2(
   const double  c2      // ikinci üyelik fonksiyonunun dönüm koordinatı
   )
```

Parametreler

c2

[in] İkinci üyelik fonksiyonunun dönüm koordinatı.

GetValue

Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

```
double  GetValue(
   const  x      // üyelik fonksiyonunun argümanı
   )
```

Parametreler

x

[in] Üyelik fonksiyonunun argümanı.

Dönüş Değeri

Üyelik fonksiyonunun değeri

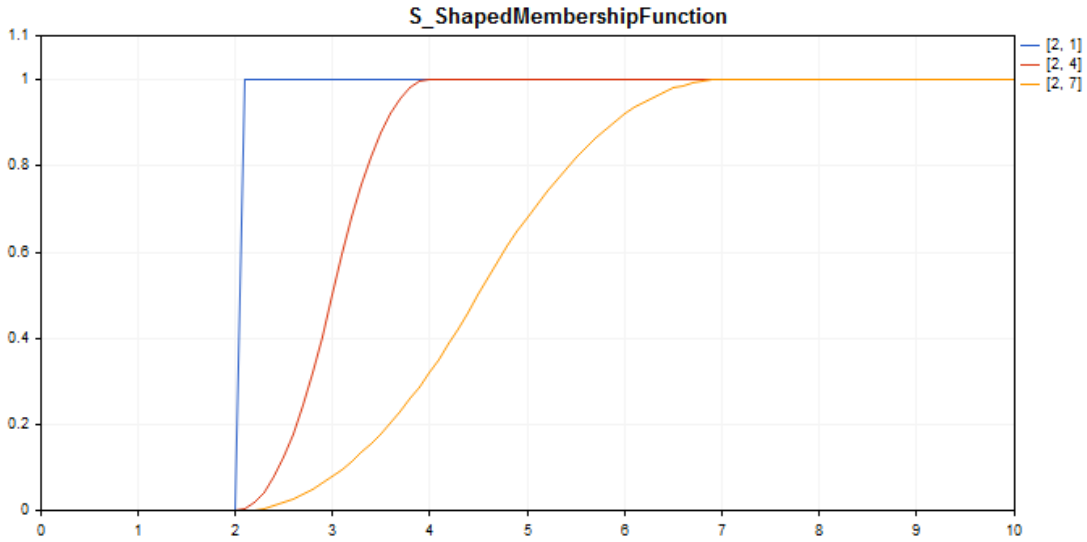
CS_ShapedMembershipFunction

A ve B parametrelerine sahip S-şekilli üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır.

Açıklama

Fonksiyon, S-şekilli iki parametreli bir üyelik fonksiyonunu ayarlar. Bu, 0 - 1 aralığında değerler alan ve azalmayan bir üyelik fonksiyonudur. Fonksiyonun parametreleri bir aralığı tanımlar. Bu aralıkta fonksiyon doğrusal olmayan şekilde 0'dan 1'e doğru artar.

Fonksiyon, çok düşük tipli bulanık kümeleri temsil eder (başka bir deyişle, azalmayan ve saturasyonlu üyelik fonksiyonlarını ayarlar).



Çizelge çizmek için oluşturulmuş bir örnek kod aşağıda verilmiştir.

Bildirim

```
class CS_ShapedMembershipFuncion : public IMembershipFunction
```

Başlık

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Kalıtım hiyerarşisi

CObject
IMembershipFunction
CS_ShapedMembershipFunction

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
A	Artış aralığının başlangıç değerini gösteren parametreyi alır/ayarlar.

Sınıf yöntemi	Açıklama
B	Bulanık küme çekirdeğinin ilk parametresini alır/ayarlar.
GetValue	Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Örnek

```
//+------------------------------------------------------------------+
//|                                  S_ShapedMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Üyelik fonksiyonlarını oluştur
CS_ShapedMembershipFunction func1(2,1);
CS_ShapedMembershipFunction func2(2,4);
CS_ShapedMembershipFunction func3(2,7);
//--- üyelik fonksiyonları için örtüler oluştur
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)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- grafiği oluştur
   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);
//--- eğri oluştur
   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]");
//--- X-ekseninin özelliklerini ayarla
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y-ekseninin özelliklerini ayarla
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- çiz
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Get yöntemi)

Artış aralığının başlangıç değerini gösteren parametreyi alır.

```
double  A()
```

Dönüş Değeri

Artış aralığının başlangıç değerini gösteren parametre.

A (Set yöntemi)

Artış aralığının başlangıç değerini gösteren parametreyi ayarlar.

```
void  A(
   const double  a      // artış aralığının başlangıç değerini gösteren parametre
   )
```

Parametreler

a

[in] Artış aralığının başlangıç parametresi.

B (Get yöntemi)

Bulanık küme çekirdeğinin ilk parametresini alır.

```
double  B()
```

Dönüş Değeri

Bulanık küme çekirdeğinin ilk parametresi.

B (Set yöntemi)

Bulanık küme çekirdeğinin ilk parametresini ayarlar.

```
void  B(
   const double  b      // bulanık küme çekirdeğinin ilk parametresi
   )
```

Parametreler

b

[in] Bulanık küme çekirdeğinin ilk parametresi.

GetValue

Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

```
double  GetValue(
   const  x      // üyelik fonksiyonunun argümanı
   )
```

Parametreler

x

[in] Üyelik fonksiyonunun argümanı.

Dönüş Değeri

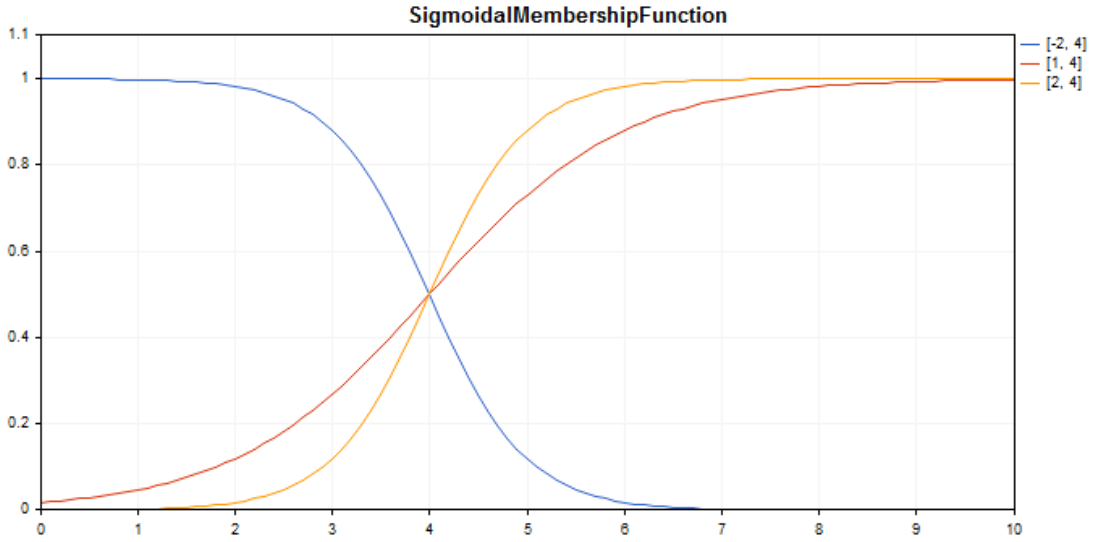
Üyelik fonksiyonunun değeri

CSigmoidalMembershipFunction

A ve C parametrelerine sahip sigmoid üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır

Açıklama

İki sigmoid üyelik fonksiyonları, monoton üyelik fonksiyonlarıyla çalışmak için kullanılır Argüman değeriyle başlayan ve değeri bire eşit olan üyelik fonksiyonlarının oluşturulmasını sağlar. Bu tip fonksiyonlar "kısa" veya "uzun" gibi sözel terimlerle çalışmak istediğinizde kullanılabilir.



Çizelge çizmek için oluşturulmuş bir örnek kod aşağıda verilmiştir.

Bildirim

```
class CSigmoidalMembershipFuncion : public IMembershipFunction
```

Başlık

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Kalıtım hiyerarşisi

CObject

IMembershipFunction

CSigmoidalMembershipFunction

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
A	Üyelik fonksiyonunun eğim oranını alır/ayarlar.
C	Üyelik fonksiyonunun dönüm koordinatı parametresini alır/ayarlar.

Sınıf yöntemi	Açıklama
GetValue	Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Örnek

```
//+------------------------------------------------------------------+
//|                                  SigmoidalMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Üyelik fonksiyonlarını oluştur
CSigmoidalMembershipFunction func1(-2, 4);
CSigmoidalMembershipFunction func2(1, 4);
CSigmoidalMembershipFunction func3(2, 4);
//--- üyelik fonksiyonları için örtüler oluştur
double SigmoidalMembershipFunction1(double x) { return(func1.GetValue(x)); }
double SigmoidalMembershipFunction2(double x) { return(func2.GetValue(x)); }
double SigmoidalMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- grafiği oluştur
   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);
//--- eğri oluştur
   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]");
//--- X-ekseninin özelliklerini ayarla
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y-ekseninin özelliklerini ayarla
```

```
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- çiz
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Get yöntemi)

Üyelik fonksiyonunun eğim oranını alır.

```
double  A()
```

Dönüş Değeri

Üyelik fonksiyonu eğim oranı.

A (Set yöntemi)

Üyelik fonksiyonunun eğim oranını ayarlar.

```
void  A(
   const double  a      // ilk üyelik fonksiyonu eğim oranı
   )
```

Parametreler

a

[in] Üyelik fonksiyonu eğim oranı.

C (Get yöntemi)

Üyelik fonksiyonunun dönüm koordinatı parametresini alır.

```
double  C()
```

Dönüş Değeri

Üyelik fonksiyonunun dönüm koordinatı parametresi.

C (Set yöntemi)

Üyelik fonksiyonunun dönüm koordinatını ayarlar.

```
void  C(
   const double  c      // üyelik fonksiyonunun dönüm koordinatı
   )
```

Parametreler

c

[in] Üyelik fonksiyonunun dönüm koordinatı.

GetValue

Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

```
double  GetValue(
   const  x      // üyelik fonksiyonunun argümanı
   )
```

Parametreler

x

[in] Üyelik fonksiyonunun argümanı.

Dönüş Değeri

Üyelik fonksiyonunun değeri

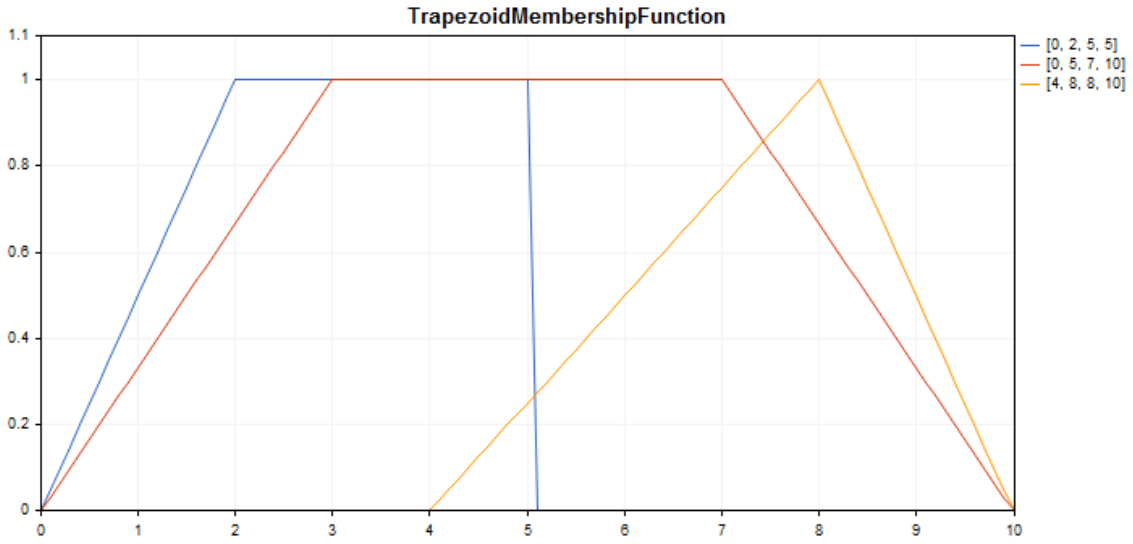
CTrapezoidMembershipFunction

X1, X2, X3 ve X4 parametrelerine sahip trapezoit üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır.

Açıklama

Fonksiyon, parçalı doğrusal yakınsama ile oluşturulur. Bu, üçgen fonksiyonlar için bir genelleştirmedir ve bulanık küme çekirdeğini bir aralık olarak atayabilmenizi sağlar. Bu tarz üyelik fonksiyonları iyimser/kötümser değerlendirmelerin uygun şekilde yorumlanabilmesini sağlar.

Fonksiyon, değişkenlerin asimetrik üyelik fonksiyonlarını ayarlamak için kullanılır (belli bir aralık içinde tanımlanan en kritik değerleri ile).



Çizelge çizmek için oluşturulmuş bir örnek kod aşağıda verilmiştir.

Bildirim

```
class CTrapezoidMembershipFuncion : public IMembershipFunction
```

Başlık

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Kalıtım hiyerarşisi

CObject

IMembershipFunction

CTrapezoidMembershipFunction

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
X1	İlk noktanın X eksenindeki değerini alır/ayarlar.

Sınıf yöntemi	Açıklama
X2	İkinci noktanın X eksenindeki değerini alır/ayarlar.
X3	Üçüncü noktanın X eksenindeki değerini alır/ayarlar.
X4	Dördüncü noktanın X eksenindeki değerini alır/ayarlar.
GetValue	Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Örnek

```
//+------------------------------------------------------------------+
//|                                  TrapezoidMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Üyelik fonksiyonlarını oluştur
CTrapezoidMembershipFunction func1(0,2,5,5);
CTrapezoidMembershipFunction func2(0,3,7,10);
CTrapezoidMembershipFunction func3(4,8,8,10);
//--- üyelik fonksiyonları için örtüler oluştur
double TrapezoidMembershipFunction1(double x) { return(func1.GetValue(x)); }
double TrapezoidMembershipFunction2(double x) { return(func2.GetValue(x)); }
double TrapezoidMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- grafiği oluştur
   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);
//--- eğri oluştur
   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
//--- X-ekseninin özelliklerini ayarla
```

```
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y-ekseninin özelliklerini ayarla
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- çiz
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

X1 (Get yöntemi)

İlk noktanın X eksenindeki değerini alır.

```
double  X1()
```

Dönüş Değeri

İlk noktanın X eksenindeki değeri.

X1 (Set yöntemi)

İlk noktanın X eksenindeki değerini ayarlar.

```
void  X1(
   const double  x1      // ilk noktanın X eksenindeki değeri
   )
```

Parametreler

x1

[in] İlk noktanın X eksenindeki değeri.

X2 (Get yöntemi)

İkinci noktanın X eksenindeki değerini alır.

```
double  X2()
```

Dönüş Değeri

İkinci noktanın X eksenindeki değeri.

X2 (Set yöntemi)

İkinci noktanın X eksenindeki değerini ayarlar.

```
void  X2(
   const double  x2      // ikinci noktanın X eksenindeki değeri
```

```
   )
```

Parametreler

x2

[in] İkinci noktanın X eksenindeki değeri.

X3 (Get yöntemi)

Üçüncü noktanın X eksenindeki değerini alır.

```
double  X3()
```

Dönüş Değeri

Üçüncü noktanın X eksenindeki değeri.

X3 (Set yöntemi)

Üçüncü noktanın X eksenindeki değerini ayarlar.

```
void  X3(
   const double  x3      // üçüncü noktanın X eksenindeki değeri
   )
```

Parametreler

x3

[in] Üçüncü noktanın X eksenindeki değeri.

X4 (Get yöntemi)

Dördüncü noktanın X eksenindeki değerini alır.

```
double  X4()
```

Dönüş Değeri

Dördüncü noktanın X eksenindeki değeri.

X4 (Set yöntemi)

Dördüncü noktanın X eksenindeki değerini ayarlar.

```
void  X4(
   const double  x4      // dördüncü noktanın X eksenindeki değeri
   )
```

Parametreler

x4

[in] Dördüncü noktanın X eksenindeki değeri.

GetValue

Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

```
double  GetValue(
   const  x      // üyelik fonksiyonunun argümanı
   )
```

Parametreler

x

[in] Üyelik fonksiyonunun argümanı.

Dönüş Değeri

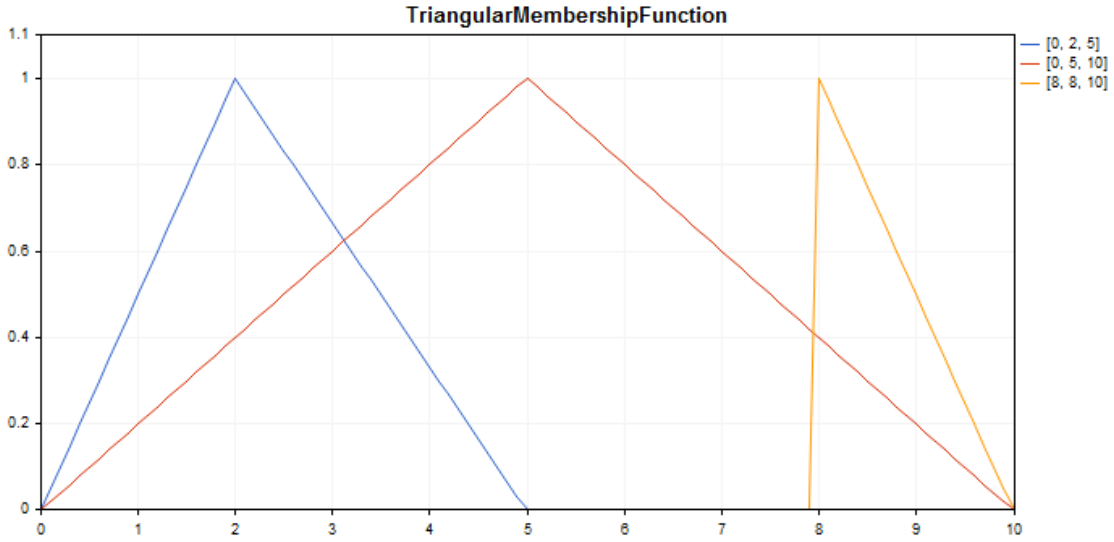
Üyelik fonksiyonunun değeri

CTriangularMembershipFunction

X1, X2 ve X3 parametrelerine sahip üçgen üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır.

Açıklama

Fonksiyon, bir üyelik fonksiyonunu üçgen biçiminde ayarlar. Bu en basit olan ve en sık uygulanan üyelik fonksiyonu biçimidir.



Çizelge çizmek için oluşturulmuş bir örnek kod aşağıda verilmiştir.

Bildirim

```
class CTriangularMembershipFuncion : public IMembershipFunction
```

Başlık

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Kalıtım hiyerarşisi

CObject
  IMembershipFunction
    CTriangularMembershipFunction

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
X1	İlk noktanın X eksenindeki değerini alır.
X2	İkinci noktanın X eksenindeki değerini alır.
X3	Üçüncü noktanın X eksenindeki değerini alır.

Sınıf yöntemi	Açıklama
ToNormalMF	Bir üçgen üyelik fonksiyonunu Gaussyen fonksiyona dönüştürür.
GetValue	Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Örnek

```
//+------------------------------------------------------------------+
//|                                TriangularMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             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>
//--- Üyelik fonksiyonlarını oluştur
CTriangularMembershipFunction func1(0,2,5);
CTriangularMembershipFunction func2(0,5,10);
CTriangularMembershipFunction func3(8,8,10);
//--- üyelik fonksiyonları için örtüler oluştur
double TriangularMembershipFunction1(double x) { return(func1.GetValue(x)); }
double TriangularMembershipFunction2(double x) { return(func2.GetValue(x)); }
double TriangularMembershipFunction3(double x) { return(func3.GetValue(x)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- grafiği oluştur
   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);
//--- eğri oluştur
   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]
```

```
//--- X-ekseninin özelliklerini ayarla
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y-ekseninin özelliklerini ayarla
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- çiz
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

X1 (Get yöntemi)

İlk noktanın X eksenindeki değerini alır.

```
double  X1()
```

Dönüş Değeri

İlk noktanın X eksenindeki değeri.

X1 (Set yöntemi)

İlk noktanın X eksenindeki değerini ayarlar.

```
void  X1(
   const double  x1      // ilk noktanın X eksenindeki değeri
   )
```

Parametreler

x1

[in] İlk noktanın X eksenindeki değeri.

X2 (Get yöntemi)

İkinci noktanın X eksenindeki değerini alır.

```
double  X2()
```

Dönüş Değeri

İkinci noktanın X eksenindeki değeri.

X2 (Set yöntemi)

İkinci noktanın X eksenindeki değerini ayarlar.

```
void  X2(
```

```
   const double   x2      // ikinci noktanın X eksenindeki değeri
   )
```

Parametreler

x2

[in] İkinci noktanın X eksenindeki değeri.

X3 (Get yöntemi)

Üçüncü noktanın X eksenindeki değerini alır.

```
double  X3()
```

Dönüş Değeri

Üçüncü noktanın X eksenindeki değeri.

X3 (Set yöntemi)

Üçüncü noktanın X eksenindeki değerini ayarlar.

```
void  X3(
   const double   x3      // üçüncü noktanın X eksenindeki değeri
   )
```

Parametreler

x3

[in] Üçüncü noktanın X eksenindeki değeri.

ToNormalMF

Bir üçgen üyelik fonksiyonunu Gaussyen fonksiyona dönüştürür.

```
CNormalMembershipFunction*  ToNormalMF()
```

Dönüş Değeri

The pointer to a Gaussian membership function.

GetValue

Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

```
double  GetValue(
   const  x      // üyelik fonksiyonunun argümanı
   )
```

Parametreler

x

[in] Üyelik fonksiyonunun argümanı.

Dönüş Değeri

Üyelik fonksiyonunun değeri

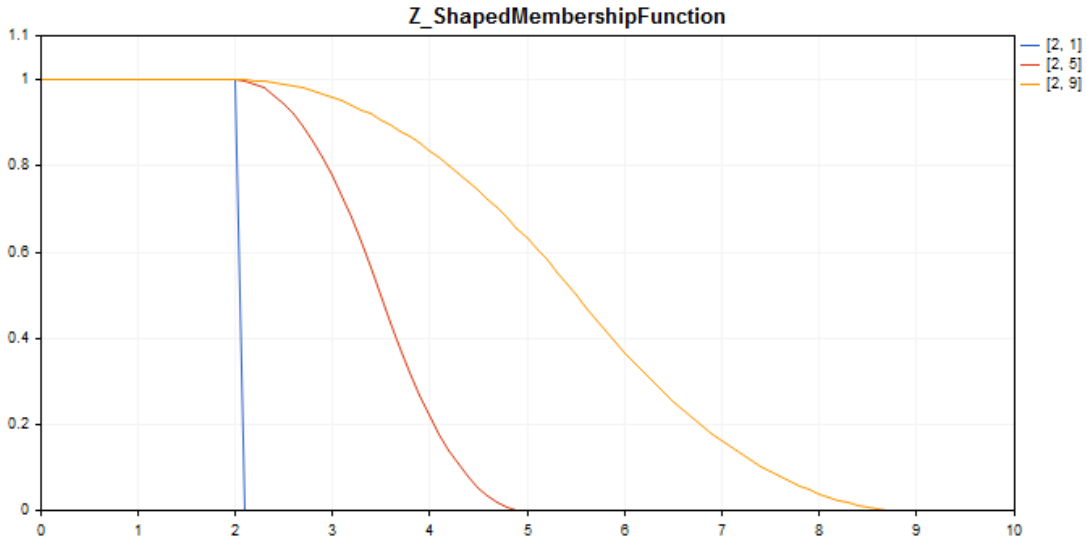
CZ_ShapedMembershipFunction

A ve B parametrelerine sahip z-şekilli üyelik fonksiyonlarıyla çalışmak için tasarlanmıştır

Açıklama

Fonksiyon, z-şekilli iki parametreli bir üyelik fonksiyonunu ayarlar. Bu, 1 - 0 aralığında değerler alan ve artmayan bir üyelik fonksiyonudur. Fonksiyonun parametreleri bir aralığı tanımlar. Bu aralıkta fonksiyon doğrusal olmayan şekilde 1 'den 0'a doğru azalır.

Fonksiyon, çok düşük tipli bulanık kümeleri temsil eder. Başka bir deyişle, artmayan ve saturasyonlu üyelik fonksiyonlarını ayarlar.



Çizelge çizmek için oluşturulmuş bir örnek kod aşağıda verilmiştir.

Bildirim

```
class CZ_ShapedMembershipFuncion : public IMembershipFunction
```

Başlık

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Kalıtım hiyerarşisi

CObject
  IMembershipFunction
    CZ_ShapedMembershipFunction

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
A	Azalma aralığının başlangıç değerini gösteren parametreyi alır/ayarlar.

Sınıf yöntemi	Açıklama
B	Azalma aralığının bitiş değerini gösteren parametreyi alır/ayarlar.
GetValue	Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Örnek

```
//+------------------------------------------------------------------+
//|                                  Z_ShapedMembershipFunction.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Math\Fuzzy\membershipfunction.mqh>
#include <Graphics\Graphic.mqh>
//--- Üyelik fonksiyonlarını oluştur
CZ_ShapedMembershipFunction func1(2,1);
CZ_ShapedMembershipFunction func2(2,5);
CZ_ShapedMembershipFunction func3(2,9);
//--- üyelik fonksiyonları için örtüler oluştur
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)); }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
//--- grafiği oluştur
   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);
//--- eğri oluştur
   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]");
//--- X-ekseninin özelliklerini ayarla
   graphic.XAxis().AutoScale(false);
   graphic.XAxis().Min(0.0);
```

```
   graphic.XAxis().Max(10.0);
   graphic.XAxis().DefaultStep(1.0);
//--- Y-ekseninin özelliklerini ayarla
   graphic.YAxis().AutoScale(false);
   graphic.YAxis().Min(0.0);
   graphic.YAxis().Max(1.1);
   graphic.YAxis().DefaultStep(0.2);
//--- çiz
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

A (Get yöntemi)

Azalma aralığının başlangıç değerini gösteren parametreyi alır.

```
double  A()
```

Dönüş Değeri

Azalma aralığının başlangıç değerini gösteren parametre.

A (Set yöntemi)

Azalma aralığının başlangıç değerini gösteren parametreyi ayarlar.

```
void  A(
   const double  a      // azalma aralığının başlangıç değerini gösteren parametre
   )
```

Parametreler

a

[in] Azalma aralığının başlangıç değerini gösteren parametre.

B (Get yöntemi)

Azalma aralığının bitiş değerini gösteren parametreyi alır.

```
double  B()
```

Dönüş Değeri

Azalma aralığının bitiş değerini gösteren parametre.

B (Set yöntemi)

Azalma aralığının bitiş değerini gösteren parametreyi ayarlar.

```
void  B(
   const double  b      // azalma aralığının bitiş değerini gösteren parametre
   )
```

Parametreler

b

[in] Azalma aralığının bitiş değerini gösteren parametre.

GetValue

Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

```
double  GetValue(
   const  x      // üyelik fonksiyonunun argümanı
   )
```

Parametreler

x

[in] Üyelik fonksiyonunun argümanı.

Dönüş Değeri

Üyelik fonksiyonunun değeri

IMembershipFunction

Tüm üyelik fonksiyonu sınıfları için temel sınıf.

Bildirim

```
class CZ_ShapedMembershipFuncion : public IMembershipFunction
```

Başlık

```
#include <Math\Fuzzy\membershipfunction.mqh>
```

Kalıtım hiyerarşisi

CObject

IMembershipFunction

İlk nesil

CCompositeMembershipFunction, CConstantMembershipFunction, CDifferencTwoSigmoidalMembershipFunction, CGeneralizedBellShapedMembershipFunction, CNormalCombinationMembershipFunction, CNormalMembershipFunction, CP_ShapedMembershipFunction, CProductTwoSigmoidalMembershipFunctions, CS_ShapedMembershipFunction, CSigmoidalMembershipFunction, CTrapezoidMembershipFunction, CTriangularMembershipFunction, CZ_ShapedMembershipFunction

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
GetValue	Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

GetValue

Üyelik fonksiyonunun değerini belli bir argümana göre hesaplar.

```
double  GetValue(
   const  x      // üyelik fonksiyonunun argümanı
   )
```

Parametreler

x

[in] Üyelik fonksiyonunun argümanı.

Dönüş Değeri

Üyelik fonksiyonunun değeri

Bulanık sistem kuralları

Bulanık sistem (bulanık mantıksal çıkarım sistemi), **bulanık kuralların** ve bulanık işlemlerin kullanımıyla, sonucun girdi ve çıktı ifadelerinin mevcut durumuna karşılık gelen bir bulanık küme şeklinde elde edilmesidir.

Bulanık kurallar incelenen nesnenin girdi ve çıktıları arasındaki ilişkiyi belirler. Sistemdeki kuralların sayısı sınırsızdır. Bulanık kuralların genel çerçevesi şu şekildedir:

Eğer *kural durumu* ise, **o zaman** *kural sonucudur*.

Kural durumu nesnenin mevcut durumunu tanımlar. *Kural sonucu*, kural durumunun nesneyi nasıl etkilediğini belirtir.

Bulanık sistemler için kurallar sınıfı	Açıklama
CMamdaniFuzzyRule	Mamdani algoritması ile bir bulanık mantık kuralını kullanmak için tasarlanmıştır
CSugenoFuzzyRule	Suheno algoritması ile bir bulanık mantık kuralını kullanmak için tasarlanmıştır
CSingleCondition	Bu sınıf "Bulanık değişken — Bulanık terim" çiftiyle tanımlanan bir bulanık durum ayarlar.
CConditions	Bu sınıf birbirine bir operatör ile bağlı olan bulanık durumların bir kümesini tanımlar.
CGenericFuzzyRule	Bulanık kuralların her iki tipi ile çalışmak için tasarlanmış temel sınıf.

CMamdaniFuzzyRule

Mamdani sistemi bulanık mantıksal çıkarımın iki temel tipinden biridir. Çıktı değişkenlerinin değerleri bulanık terimlerle ayarlanır.

Açıklama

Mamdani algoritmasının bulanık mantık kuralı şu şekilde tanımlanır:

$$if(X_1 \ is \ a_1) \wedge (X_2 \ is \ a_2) \wedge \ldots \wedge (X_n \ is \ a_n) \ then \ (Y \ is \ d)(W)$$

burada:

- X = (X1, X2, X3 ... Xn) — girdi değişkenleri vektörü;
- Y — çıktı değişkeni;
- a = (a1, a2, a3 ... an) — girdi değişkeni değerlerinin vektörü;
- d — çıktı değişkeninin değeri;
- W — kural ağırlığı.

Bildirim

```
class CMamdaniFuzzyRule : public CGenericFuzzyRule
```

Başlık

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Kalıtım hiyerarşisi

CObject
  IParsableRule
    CGenericFuzzyRule
      CMamdaniFuzzyRule

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
Conclusion	Bulanık Mamdani kuralının sonucunu alır/ayarlar
Weight	Bulanık Mamdani kuralının ağırlığını alır/ayarlar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CGenericFuzzyRule

Condition, Condition, CreateCondition, CreateCondition, CreateCondition

Conclusion (Get yöntemi)

Bulanık Mamdani kuralının sonucunu alır.

```
CSingleConditon*  Conclusion()
```

Dönüş Değeri

Mamdani kuralının sonucu.

Conclusion (Set yöntemi)

Bulanık Mamdani kuralının sonucunu ayarlar.

```
void  Conclusion(
   CSingleConditon*  value      // bulanık Mamdani kuralının sonucu
   )
```

Parametreler

value

[in] Bulanık Mamdani kuralının sonucu.

Weight (Get yöntemi)

Bulanık Mamdani kuralının ağırlığını alır.

```
double  Weight()
```

Dönüş Değeri

Bulanık Mamdani kuralının ağırlığı.

Weight (Set yöntemi)

Bulanık Mamdani kuralının ağırlığını ayarlar.

```
void  Weight(
   const double  value      // bulanık Mamdani kuralının ağırlığı
   )
```

Parametreler

value

[in] Bulanık Mamdani kuralının ağırlığı.

CSugenoFuzzyRule

Sugeno sistemi bulanık mantıksal çıkarımın iki temel tipinden biridir. Çıktı değişkeninin değerleri girdi değişkenlerinin bir doğrusal kombinasyonu şeklinde ayarlanır.

Açıklama

Mamdani kuralının aksine, girdi değişkeninin değeri bir bulanık terimle değil, bir doğrusal fonksiyonla belirlenir. Sugeno algoritmasının bulanık mantık kuralı şu şekilde tanımlanır:

$$if(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)\ then\ (Y=b_0+b_1\cdot X_1+b_2\cdot X_2+\ldots+b_n\cdot X_n)(W)$$

burada:

- X = (X1, X2, X3 ... Xn) — girdi değişkenleri vektörü;
- Y — çıktı değişkeni;
- a = (a1, a2, a3 ... an) — girdi değişkeni değerlerinin vektörü;
- b = (b1, b2, b3 ... bn) — çıktı değeri için doğrusal fonksiyonun serbest teriminin oranı
- W — kural ağırlığı.

Bildirim

```
class CSugenoFuzzyRule : public CGenericFuzzyRule
```

Başlık

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Kalıtım hiyerarşisi

CObject

IParsableRule

CGenericFuzzyRule

CSugenoFuzzyRule

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
Conclusion	Bulanık Sugeno kuralının sonucunu alır/ayarlar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CGenericFuzzyRule

Condition, Condition, CreateCondition, CreateCondition, CreateCondition

Conclusion (Get yöntemi)

Bulanık Sugeno kuralının sonucunu alır.

```
CSingleConditon*  Conclusion()
```

Dönüş Değeri

Bulanık Sugeno kuralının sonucu.

Conclusion (Set yöntemi)

Bulanık Sugeno kuralının sonucunu ayarlar.

```
void  Conclusion(
   CSingleCondition*  value      // bulanık Sugeno kuralının sonucu
   )
```

Parametreler

value

[in] Bulanık Sugeno kuralının sonucu.

CSingleCondition

Bu sınıf "Bulanık değişken — Bulanık terim" çiftiyle tanımlanan bir bulanık durum ayarlar.

Açıklama

Bulanık duruma göre, bir değişken bir terime karşılık gelir. Bir bulanık durum şu ifade ile açıklanabilir: *X, a'dır,*

burada:

- X bir bulanık değişkendir;
- a , bulanık değişken değeridir (bulanık terim).

Bildirim

```
class CSingleCondition : public ICondition
```

Başlık

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Kalıtım hiyerarşisi

CObject

ICondition

CSingleCondition

İlk nesil

CFuzzyCondition

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
Not	Bu duruma olumsuzluk uygulanmasının gerekliliğini belirten bayrağın değerini alır/ayarlar
Term	Bu durum için bir bulanık terim alır/ayarlar
Var	Bu durum için bir bulanık değişken alır/ayarlar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Not (Get yöntemi)

Bu durumlara olumsuzluk uygulanmasının gerekliliğini belirten bayrağın değerini alır.

```
bool Not()
```

Dönüş Değeri

Bayrak değeri.

Not (Set yöntemi)

Bu durumlara olumsuzluk uygulanmasının gerekliliğini belirten bayrağın değerini ayarlar.

```
void  Not(
   bool  not      // bayrak değeri
   )
```

Parametreler

not

[in] Bayrak değeri.

Term (Get yöntemi)

Verilen durum için bir bulanık terim alır.

```
INamedValue* Term()
```

Dönüş Değeri

Verilen durum için bir bulanık terim.

Term (Set yöntemi)

Verilen durum için bir bulanık terim ayarlar.

```
void  Term(
   INamedValue*&  value      // verilen durum için bulanık terim
   )
```

Parametreler

value

[in] Verilen durum için bir bulanık terim.

Var (Get yöntemi)

Verilen durum için bir bulanık değişken alır.

```
INamedVariable* Var()
```

Dönüş Değeri

Verilen durum için bir bulanık değişken.

Var (Set yöntemi)

Verilen durum için bir bulanık değişken ayarlar.

```
void  Var(
   INamedVariable*&  value      // verilen durum için bir bulanık değişken
```

```
)
```

Parametreler

value

[in] bulanık değişken.

CConditions

Bu sınıf birbirine bir operatör ile bağlı olan bulanık durumların bir kümesini tanımlar.

Açıklama

Birbirine bir operatör ile bağlı olan bulanık durumların bir kümesi şu şekilde açıklanabilir:

$$(X_1 \; is \; a_1) \wedge (X_2 \; is \; a_2) \wedge \ldots \wedge (X_n \; is \; a_n)$$

burada:

- X = (X1, X2, X3 ... Xn) — girdi değişkenlerinin vektörü;
- a = (a1, a2, a3 ... an) — girdi değişkenlerinin değerlerinin vektörü.

Bu örnekte *ve* operatörü kullanılmıştır. Sınıf dahilinde *veya* operatörü de bulunmaktadır.

Bildirim

```
class CConditions : public ICondition
```

Başlık

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Kalıtım hiyerarşisi

CObject
  ICondition
    CConditions

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
ConditionsList	Tüm durumların listesini alır.
Not	Bu durumlara olumsuzluk uygulanmasının gerekliliğini belirten bayrağın değerini alır/ayarlar
Op	Durum paketi operatörünün tipini alır/ayarlar.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

ConditionsList

Tüm durumların listesini alır.

```
CList*  ConditionsList()
```

Dönüş Değeri

Tüm durumların listesi.

Not (Get yöntemi)

Bu durumlara olumsuzluk uygulanmasının gerekliliğini belirten bayrağın değerini alır.

```
bool Not()
```

Dönüş Değeri

Bayrak değeri.

Not (Set yöntemi)

Bu durumlara olumsuzluk uygulanmasının gerekliliğini belirten bayrağın değerini ayarlar

```
void  Not(
   bool  not      // bayrak değeri
   )
```

Parametreler

not

[in] Bayrak değeri.

Op (Get yöntemi)

Durum paketi operatörünün tipini alır. *ve* ve *veya* operatörleri mevcuttur.

```
OperatorType Op()
```

Dönüş Değeri

Durum paketi operatörünün tipi.

Op (Set yöntemi)

Durum paketi operatörünün tipini ayarlar. *ve* ve *veya* operatörleri mevcuttur.

```
void  Op(
   OperatorType  op     // durum paketi operatörünün tipi
   )
```

Parametreler

op

[in] Durum paketi operatörünün tipi.

CGenericFuzzyRule

Bulanık kuralların her iki tipi ile çalışmak için tasarlanmış temel sınıf.

Bildirim

```
class CGenericFuzzyRule : public IParsableRule
```

Başlık

```
#include <Math\Fuzzy\fuzzyrule.mqh>
```

Kalıtım hiyerarşisi

CObject

IParsableRule

CGenericFuzzyRule

İlk nesil

CMamdaniFuzzyRule, CSugenoFuzzyRule

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
Conclusion	Bulanık kuralın sonucunu alır/ayarlar
Condition	Bir bulanık kural için 'eğer' durumunu (veya durumlar kümesini) alır/ayarlar.
CreateCondition	Belirtilen parametrelere göre bulanık kural için bir durum oluşturur

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Conclusion (Get yöntemi)

Bulanık kuralın sonucunu alır.

```
CSingleConditon*  Conclusion()
```

Dönüş Değeri

Bulanık kuralın sonucu.

Conclusion (Set yöntemi)

Bulanık kuralın sonucunu ayarlar.

```
virtual void  Conclusion(
   CSingleConditon*  value      // bulanık kuralın sonucu
```

```
   )
```

Parametreler

value

[in] Bulanık kuralın sonucu.

Condition (Get yöntemi)

Bir bulanık kural için 'eğer' durumunu (veya durumlar kümesini) alır.

```
CConditons*  Condition()
```

Dönüş Değeri

Bulanık durum (veya durumlar kümesi).

Condition (Set yöntemi)

Bir bulanık kural için 'eğer' durumunu (veya durumlar kümesini) ayarlar.

```
void  Condition(
   CConditons*  value      // bir bulanık kural için 'eğer' durumu (veya durumlar küme
   )
```

Parametreler

value

[in] Bulanık durum (veya durumlar kümesi).

CreateCondition

Belirtilen parametrelere göre bulanık kural için bir durum oluşturur.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable*  var,       // bulanık değişken
   CFuzzyTerm*      term,      // bulanık terim
   )
```

Parametreler

var

[in] Bulanık değişken.

term

[in] Bulanık terim.

Dönüş Değeri

Bulanık kural durumu.

CreateCondition

Belirtilen parametrelere göre bulanık kural için bir durum oluşturur.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable*  var,      // bulanık değişken
   CFuzzyTerm*      term,     // bulanık terim
   bool             not,      // duruma olumsuzluk ekleme gerekliliğini gösteren bayra
   )
```

Parametreler

var

[in] Bulanık değişken.

term

[in] Bulanık terim.

not

[in] Duruma olumsuzluk ekleme gerekliliğini gösteren bayrak.

Dönüş Değeri

Bulanık kural durumu.

CreateCondition

Belirtilen parametrelere göre bulanık kural için bir durum oluşturur.

```
CFuzzyCondition*  CreateCondition(
   CFuzzyVariable* var,      // bulanık değişken
   CFuzzyTerm*     term,     // bulanık terim
   bool            not,      // duruma olumsuzluk ekleme gerekliliğini gösteren bayral
   HedgeType       hedge     // durum paketinin tipi
   )
```

Parametreler

var

[in] Bulanık değişken.

term

[in] Bulanık terim.

not

[in] Duruma olumsuzluk ekleme gerekliliğini gösteren bayrak.

hedge

[in] Durum paketinin tipi.

Dönüş Değeri

Bulanık kural durumu.

Bulanık sistem değişkenleri

Bulanık (sözel) değişkenler **bulanık sistemlerde kullanılır**. Bunlar, doğal veya yapay dillerdeki sözcük veya sözcük kombinasyonlarını değer olarak alan değişkenlerdir.

Bulanık kümeleri oluştururlar. Her bir bulanık değişkenin doğası ve sayısı, bulanık kümeler tanımlanırken belirtilen her bir görev için değişir.

Sınıf	Açıklama
CFuzzyVariable	Genel bulanık değişkenler oluşturmak için tasarlanmıştır.
CSugenoVariable	Sugeno-tipli bulanık değişkenler oluşturmak için tasarlanmıştır.

CFuzzyVariable

Genel bulanık değişkenler oluşturmak için tasarlanmıştır.

Açıklama

Burada bulanık değişken şu parametrelerle oluşturulur:

- maksimum değişken değeri;
- minimum değişken değeri;
- bulanık değişken değeri;
- terim kümesi (tüm olası değerlerin kümesi; sözel bir değişken alınması mümkündür).

Bildirim

```
class CFuzzyVariable : public CNamedVariableImpl
```

Başlık

```
#include <Math\Fuzzy\fuzzyvariable.mqh>
```

Kalıtım hiyerarşisi

CObject
  INamedValue
    INamedVariable
      CNamedVariableImpl
        CFuzzyVariable

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
AddTerm	Bulanık değişkene bir bulanık terim ekler.
GetTermByName	Belirtilen isme sahip bulanık terime dönüş yapar.
Max	Bir bulanık değişken için bir maksimum değer alır/ayarlar.
Min	Bir bulanık değişken için bir minimum değer alır/ayarlar.
Terms	Verilen bulanık değişken için bir bulanık terimler listesi alır/ayarlar.
Values	Verilen bulanık değişken için bir bulanık terimler listesi alır/ayarlar.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CNamedVariableImpl

Name, Name

AddTerm

Bulanık değişkene bir bulanık terim ekler.

```
void  AddTerm(
   CFuzzyTerm*&  term      // bulanık terim
   )
```

Parametreler

term

[in] Bulanık terim.

GetTermByName

Belirtilen isme sahip bulanık terime dönüş yapar.

```
CFuzzyTerm*  GetTermByName(
   const string  name      // bulanık terim ismi
   )
```

Parametreler

name

[in] Bulanık terim ismi.

Dönüş Değeri

Belirtilen isme sahip bulanık terim.

Max (Get yöntemi)

Bir bulanık değişken için maksimum değeri alır.

```
double  Max()
```

Dönüş Değeri

Bulanık değişkenin maksimum değeri.

Max (Set yöntemi)

Bir bulanık değişken için bir maksimum değer ayarlar.

```
void  Max(
   const double  max      // bulanık değişken için bir maksimum değer
```

```
   )
```

Parametreler

max

[in] Bulanık değişken için bir maksimum değer.

Min (Get yöntemi)

Bulanık değişkenin minimum değerini alır.

```
double  Min()
```

Dönüş Değeri

Bulanık değişkenin minimum değeri.

Max (Set yöntemi)

Bir bulanık değişken için minimum değer ayarlar.

```
void  Min(
   const double  min      // bulanık değişken için minimum değer
   )
```

Parametreler

min

[in] Bulanık değişkenin minimum değeri.

Terms (Get yöntemi)

Verilen bulanık değişken için bir bulanık terimler listesi alır.

```
CList*  Terms()
```

Dönüş Değeri

Verilen bulanık değişken için bulanık terimler listesi.

Terms (Set yöntemi)

Verilen bulanık değişken için bir bulanık terimler listesi ayarlar.

```
void  Terms(
   CList*&  terms      // bulanık değişken için bir bulanık terimler listesi
   )
```

Parametreler

terms

[in] Bulanık değişken için bir bulanık terimler listesi.

Values

Verilen bulanık değişken için bir bulanık terimler listesi alır.

```
CList*  Values()
```

Dönüş Değeri

Verilen bulanık değişken için bulanık terimler listesi.

CSugenoVariable

Sugeno-tipli bulanık değişkenler oluşturmak için tasarlanmıştır.

Açıklama

Sugeno-tipli bulanık değişkenler, bir terim yerine bir lineer fonksiyonlar kümesiyle ayarlanır ve bu açıdan genel sözel değişkenlerden farklıdır.

Bildirim

```
class CSugenoVariable : public CNamedVariableImpl
```

Başlık

```
#include <Math\Fuzzy\sugenovariable.mqh>
```

Kalıtım hiyerarşisi

CObject
  INamedValue
    INamedVariable
      CNamedVariableImpl
        CSugenoVariable

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
Functions	Bulanık Sugeno değişkeninin doğrusal fonksiyonlarının listesini alır.
GetFuncByName	Belirtilen isme sahip doğrusal fonksiyona dönüş yapar.
Values	Bulanık Sugeno değişkeninin doğrusal fonksiyonlarının listesini alır.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CNamedVariableImpl

Name, Name

Functions

Bulanık Sugeno değişkeninin doğrusal fonksiyonlarının listesini alır.

```
CList*  Functions()
```

Dönüş Değeri

Lineer fonksiyonların listesi.

GetFuncByName

Belirtilen isme sahip doğrusal fonksiyona dönüş yapar.

```
ISugenoFunction*  GetFuncByName(
   const string  name      // doğrusal fonksiyon ismi
   )
```

Parametreler

name

[in] Doğrusal fonksiyon ismi.

Dönüş Değeri

Belirtilen isme sahip doğrusal fonksiyon.

Values

Bulanık Sugeno değişkeninin doğrusal fonksiyonlarının listesini alır.

```
CList*  Values()
```

Dönüş Değeri

Bulanık Sugeno değişkeninin doğrusal fonksiyonlarının listesi.

CFuzzyTerm (bulanık terimler)

Bulanık terimleri uygulanması için tasarlanmış bir sınıftır.

Açıklama

Terim, terimler sınıfının bir elemanıdır. İki bileşenden oluşur:

- bulanık terim ismi;
- üyelik fonksiyonu.

Bildirim

```
class CFuzzyTerm : public CNamedValueImpl
```

Başlık

```
#include <Math\Fuzzy\fuzzyterm.mqh>
```

Kalıtım hiyerarşisi

CObject
  INamedValue
    CNamedValueImpl
      CFuzzyTerm

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
MembershipFunction	Bulanık terimin üyelik fonksiyonunu alır.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CNamedValueImpl

Name, Name

MembershipFunction

Bulanık terimin üyelik fonksiyonunu alır.

```
IMembershipFunction*  MembershipFunction()
```

Dönüş Değeri

Üyelik fonksiyonu

Bulanık sistemler

Bulanık sistemler (veya *bulanık modeller*) hesaplamaları bulanık mantığa dayanan matematiksel modellerdir. Bunlar, çalışma konusunun biçimselliğinin zayıf olması veya matematiksel açıklamasının aşırı karmaşık olması durumunda kurulabilir.

Model oluşturma süreci üç ana aşamaya bölünebilir:

1. Modelin girdi ve çıktı özelliklerinin tanımlanması.
2. Bilgi temelinin kurulması.
3. Bulanık çıkarım yöntemlerinden (Mamdani ve Sugeno) birinin seçilmesi.

İlk aşama, sonraki iki aşamayı da etkiler ve modelin gelecekteki işlemleri için belirleyicidir.

Bilgi tabanı (*kural tabanı*), üstünde çalışılan nesnenin girdileri ve çıktıları arasındaki bağıntıyı "eğer" ve "ise" gibi işlemlerle tanımlayan bir bulanık kurallar kümesidir.

Kural durumu nesnenin mevcut durumunu, **kural sonucu** ise durumun nesneyi nasıl etkilediğini açıklar.

Her Kural için iki tip terim ve sonuç olabilir:

1. basit (Csinglcond bağlantısı) — bir bulanık değişken içerir;
2. karmaşık (Cconditions bağlantısı) — birkaç bulanık değişken içerir.

Sistemdeki her kuralın bir **ağırlığı** vardır, bu ağırlık kuralın modeldeki önemini gösterir.. Kurallara atanan ağırlıklar [0, 1] aralığında olmalıdır.

Oluşturulan bilgi temeline göre, model için kullanılacak bulanık çıkarım sistemi belirlenir. **Bulanık mantıksal çıkarım**, bilgi tabanı ve bulanık işlemlerin kullanılmasının ardından, sonuçların, girdilerin mevcut durumunu gösteren bir bulanık küme şeklinde elde edilmesidir. Bulanık çıkarımın Mamdani ve Sugeno olmak üzere iki ana tipi mevcuttur.

Mamdani sistemi

Mamdani sisteminde çıktı değişkenlerinin değerleri bulanık terimlerle ayarlanır.

Açıklama

Mamdani algoritmasının bulanık mantık kuralı şu şekilde tanımlanır:

$$if(X_1 \ is \ a_1) \wedge (X_2 \ is \ a_2) \wedge \ldots \wedge (X_n \ is \ a_n) \ then \ (Y \ is \ d)(W)$$

burada:

- X = (X1, X2, X3 ... Xn) — girdi değişkenleri vektörü;
- Y — çıktı değişkeni;
- a = (a1, a2, a3 ... an) — girdi değişkeni değerlerinin vektörü;
- d — çıktı değişkeninin değeri;
- W — kural ağırlığı.

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
AggregationMethod	Durum paketleme tipini ayarlar.
Calculate	Sistem için bulanık çıkarımı hesaplar
DefuzzificationMethod	Bulanıklaştırma yönteminin tipini ayarlar.
EmptyRule	Mevcut sisteme göre bir boş Mamdani kuralı oluşturur.
ImplicationMethod	Sistem çıkarımı operatörünün tipini ayarlar.
Output	Bulanık Mamdani çıktı değişkenlerinin listesini alır.
OutputByName	Belirtilen isimdeki bulanık Mamdani çıktı değişkenini alır.
ParseRule	Belirtilen çizgiye göre bir bulanık Mamdani kuralı oluşturur.
Rules	Bulanık Mamdani kurallarının listesine dönüş yapar.

Sınıftan türetilen yöntemler CGenericFuzzySystem

Input, AndMethod, AndMethod, OrMethod, OrMethod, InputByName, Fuzzify

AggregationMethod

Durum toplama yönteminin tipini ayarlar.

```
void  AggregationMethod(
   AggregationMethod  value      // toplama yönteminin tipi
```

```
   )
```

Parametreler

value

[in] Durum toplama yönteminin tipi.

Calculate

Sistem için bulanık çıkarımı hesaplar

```
CList*  Calculate(
   CList*  inputValues      // girdi verisi
   )
```

Parametreler

inputValues

[in] Hesaplama için girdi verisi

Dönüş Değeri

Hesaplama sonucu.

DefuzzificationMethod

Bulanıklaştırma yönteminin tipini ayarlar.

```
void  DefuzzificationMethod(
   DefuzzificationMethod  value      // bulanıklaştırma yönteminin tipi.
   )
```

Parametreler

value

[in] Bulanıklaştırma yönteminin tipi.

EmptyRule

Mevcut sisteme göre bir boş Mamdani kuralı oluşturur.

```
CMamdaniFuzzyRule*  EmptyRule()
```

Dönüş Değeri

Bulanık Mamdani kuralı.

ImplicationMethod

Sistem çıkarımı operatörünün tipini ayarlar.

```
void  ImplicationMethod(
   ImplicationMethod  value      // çıkarım operatörünün tipi
```

```
   )
```

Parametreler

value

[in] Sistem çıkarımı operatörünün tipi.

Output

Bulanık Mamdani çıktı değişkenlerinin listesini alır.

```
CList*  Output()
```

Dönüş Değeri

Bulanık değişkenlerin listesi.

OutputByName

Belirtilen isimdeki bulanık Mamdani çıktı değişkenini alır.

```
CFuzzyVariable*  OutputByName(
   const string  name      // bulanık değişkenin ismi
   )
```

Parametreler

name

[in] Bulanık değişkenin ismi.

Dönüş Değeri

Belirtilen isimdeki bulanık Mamdani çıktı değişkeni.

ParseRule

Belirtilen çizgiye göre bir bulanık Mamdani kuralı oluşturur.

```
CMamdaniFuzzyRule*  ParseRule(
   const string  rule      // bulanık kuralın metinsel temsili
   )
```

Parametreler

rule

[in] Bulanık Mamdani kuralının dizgi şeklindeki temsili.

Dönüş Değeri

Bulanık Mamdani kuralı.

Rules

Bulanık Mamdani kurallarının listesine dönüş yapar.

```
CList*  Rules()
```

Dönüş Değeri

Bulanık Mamdani kurallarının listesi.

CSugenoFuzzyRule

Sugeno sistemi bulanık mantıksal çıkarımın iki temel tipinden biridir. Çıktı değişkeninin değerleri girdi değişkenlerinin bir doğrusal kombinasyonu şeklinde ayarlanır.

Açıklama

Mamdani kuralının aksine, girdi değişkeninin değeri bir bulanık terimle değil, bir doğrusal fonksiyonla belirlenir. Sugeno algoritmasının bulanık mantık kuralı şu şekilde tanımlanır:

$$if(X_1\ is\ a_1)\wedge(X_2\ is\ a_2)\wedge\ldots\wedge(X_n\ is\ a_n)\ then\ (Y=b_0+b_1\cdot X_1+b_2\cdot X_2+\ldots+b_n\cdot X_n)(W)$$

burada:

- X = (X1, X2, X3 ... Xn) — girdi değişkenleri vektörü;
- Y — çıktı değişkeni;
- a = (a1, a2, a3 ... an) — girdi değişkeni değerlerinin vektörü;
- b = (b1, b2, b3 ... bn) — çıktı değeri için doğrusal fonksiyonun serbest teriminin oranı
- W — kural ağırlığı.

Sınıf yöntemleri

Sınıf yöntemi	Açıklama
Calculate	Sistem için bulanık çıkarımı hesaplar
CreateSugenoFunction	Sistem için doğrusal Sugeno fonksiyonu oluşturur.
EmptyRule	Mevcut sisteme göre bir boş Sugeno kuralı oluşturur.
Output	Bulanık Sugeno çıktı değişkenlerinin listesini alır.
OutputByName	Belirtilen isimdeki bulanık Sugeno çıktı değişkenini alır.
ParseRule	Belirtilen çizgiye göre bir bulanık Sugeno kuralı oluşturur.
Rules	Bulanık kurallar listesine dönüş yapar.

Sınıftan türetilen yöntemler CGenericFuzzySystem

Input, AndMethod, AndMethod, OrMethod, OrMethod, InputByName, Fuzzify

Calculate

Sistem için bulanık çıkarımı hesaplar

```
CList*  Calculate(
   CList*&  inputValues      // girdi verisi
   )
```

Parametreler

inputValues

[in] Hesaplama için girdi verisi

Dönüş Değeri

Hesaplama sonucu.

CreateSugenoFunction

Sistem için doğrusal Sugeno fonksiyonu oluşturur.

```
CLinearSugenoFunction*  CreateSugenoFunction(
   const string  name,           // fonksiyon ismi
   const double& coeffs[]        // fonksiyon oranları
   )
```

Parametreler

name

[in] Fonksiyon ismi.

coeffs[]

[in] Fonksiyon oranları.

Dönüş Değeri

Doğrusal Sugeno fonksiyonu.

Not

Oran dizisinin boyutu girdi sayısı veya bir fazlası kadar olabilir. İlk durumda Sugeno doğrusal fonksiyonunun serbest terimi sıfıra eşitken, ikinci durumda son orana eşittir.

CreateSugenoFunction

Sistem için doğrusal Sugeno fonksiyonu oluşturur.

```
CLinearSugenoFunction*  CreateSugenoFunction(
   const string  name,           // fonksiyon ismi
   CList*&       coeffs,         // "bulanık değişken - değişken oranı" çiftlerinin lis
   const double  constValue      // fonksiyon serbest terim oranı
   )
```

Parametreler

name

[in] Fonksiyon ismi.

coeffs[]

[in] Fonksiyon oranları.

Dönüş Değeri

Doğrusal Sugeno fonksiyonu.

EmptyRule

Mevcut sisteme göre bir boş Sugeno kuralı oluşturur.

```
CSugenoFuzzyRule*  EmptyRule()
```

Dönüş Değeri

Bulanık Sugeno kuralı.

Output

Bulanık Sugeno çıktı değişkenlerinin listesini alır.

```
CList*  Output()
```

Dönüş Değeri

Bulanık değişkenlerin listesi.

OutputByName

Belirtilen isimdeki bulanık Sugeno çıktı değişkenini alır.

```
CSugenoVariable*  OutputByName(
   const string  name      // bulanık değişkenin ismi
   )
```

Parametreler

name

Bulanık değişkenin ismi.

Dönüş Değeri

Belirtilen isimdeki bulanık Sugeno çıktı değişkeni.

ParseRule

Belirtilen çizgiye göre bir bulanık Sugeno kuralı oluşturur.

```
CSugenoFuzzyRule*  ParseRule(
   const string  rule      // bulanık Sugeno kuralının metinsel temsili
   )
```

Parametreler

rule

[in] Bulanık Sugeno kuralının metinsel temsili.

Dönüş Değeri

Bulanık Sugeno kuralı.

Rules

Bulanık kurallar listesine dönüş yapar.

```
CList*  Rules()
```

Dönüş Değeri

Bulanık kuralların listesi.

OpenCL programları ile çalışmak için tasarlanmış bir sınıftır

COpenCL class is a wrapper to facilitate working with the OpenCL functions. Bazı durumlarda, GPU kullanımı hesaplama hızını oldukça artırır.

Bu sınıfın float ve double tipli değerlere dayalı kullanımı için gereken örnekler MQL5\Scripts\Examples\OpenCL\ klasöründe bulunabilir. OpenCL programlarının kaynak kodları MQL5\Scripts\Examples\OpenCL\Double\Kernels ve MQL5\Scripts\Examples\OpenCL\Float\Kernels alt dizinlerine kaydedilir.

- MatrixMult.mq5 - global ve yerel bellek kullanımıyla matris çarpımı örneği
- BitonicSort.mq5 - dizi elemanlarının GPU ile paralel sıralanması örneği
- FFT.mq5 - Hızlı Fourier dönüşümünün hesaplanması örneği
- Wavelet.mq5 - Morlet dalgacığı ile dalgacık dönüşümü örneği.

OpenCL kaynak kodunun ayrı CL dosyalarına yazılması tavsiye edilir. Bunlar daha sonra kaynak değişkenleri yardımıyla MQL5 programına eklenebilir.

Bildirim

```
class COpenCL
```

Başlık

```
#include <OpenCL\OpenCL.mqh>
```

Sınıf yöntemleri

İsim	Açıklama
BufferCreate	Belirtilen indis üzerinde bir OpenCL tamponu oluşturur
BufferFree	Belirtilen indis üzerindeki tamponu siler
BufferFromArray	Belirtilen indis üzerinde bir değer dizisinden tampon oluşturur
BufferRead	Belirtilen indis üzerindeki OpenCL tamponu bir diziye aktarır
BufferWrite	Bir diziyi belirtilen indis üzerindeki tampona yazar
Execute	Belirtilen indis üzerindeki OpenCL kernelini çalıştırır
GetContext	OpenCL bağlamının tanıtıcı değerine dönüş yapar
GetKernel	Belirtilen indis üzerindeki kernel nesnesinin tanıtıcı değerine dönüş yapar
GetKernelName	Belirtilen indis üzerindeki kernel nesnesinin ismine dönüş yapar
GetProgram	OpenCL programının tanıtıcı değerine dönüş yapar
Initialize	OpenCL programınıbaşlatır
KernelCreate	Belirtilen indis üzerindeki OpenCL programına bir giriş noktası ekler
KernelFree	Belirtilen indis üzerindeki OpenCL start fonksiyonunu siler

İsim	Açıklama
SetArgument	Belirtilen indis üzerindeki OpenCL fonksiyonu için bir parametre ayarlar.
SetArgumentBuffer	Bir OpenCL tamponunu, belirtilen indis üzerindeki OpenCL fonksiyonunun parametresi olarak ayarlar
SetArgumentLocalMemory	Belirtilen indis üzerindeki OpenCL fonksiyonu için yerel bellekte bir parametre ayarlar.
SetBuffersCount	Tampon sayısını ayarlar
SetKernelsCount	Kernel nesnelerinin sayısını ayarlar
Shutdown	OpenCL programını kaldırır
SupportDouble	Kayan noktalı veri tiplerinin aygıt üzerinde desteklenip desteklenmediğini kontrol eder

BufferCreate

Belirtilen indis üzerinde bir OpenCL tamponu oluşturur.

```
bool  BufferCreate(
   const int   buffer_index,      // tampon indisi
   const uint  size_in_bytes,     // bayt cinsinden tampon boyutu
   const uint  flags              // tampon özelliklerini belirten bayrak kombinasyonu
   );
```

Parametreler

buffer_index

[in] Indis tamponu.

size_in_bytes

[in] Tampon büyüklüğü.

flags

[in] Tampon özelliklerini belirten bayrak kombinasyonu.

Dönüş Değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

BufferFree

Belirtilen indis üzerindeki tamponu siler.

```
bool  BufferFree(
   const int  buffer_index      // tampon indisi
   );
```

Parametreler

buffer_index

[in] Indis tamponu.

Dönüş Değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

BufferFromArray

Belirtilen indis üzerinde bir değer dizisinden tampon oluşturur.

```
template<typename T>
bool  BufferFromArray(
   const int   buffer_index,          // tampon indisi
   T           &data[],               // değer dizisi
   const uint  data_array_offset,     // bayt cinsinden yazma başlangıç konumu
   const uint  data_array_count,      // yazılacak dizi elemanlarının sayısı
   const uint  flags                  // tampon özelliklerini belirten bayrak kombinas
   );
```

Parametreler

buffer_index

[in] Indis tamponu.

&data[]

[in] OpenCL tamponuna yazılacak verilerden oluşan dizi.

data_array_offset

[in] OpenCL tamponu içinde bayt bazında yazma başlangıç konumu.

data_array_count

[in] Yazılacak değerlerin sayısı.

flags

[in] Tampon özelliklerini belirten bayrak kombinasyonu.

Dönüş Değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

BufferRead

Belirtilen indis üzerindeki OpenCL tamponu bir diziye aktarır.

```
template<typename T>
bool  BufferRead(
   const int   buffer_index,          // tampon indisi
   T           &data[],               // değer dizisi
   const uint  cl_buffer_offset,      // OpenCL tamponunun başlangıç konumu (bayt cins
   const uint  data_array_offset,     // yazılacak ilk dizi elemanının indisi
   const uint  data_array_count       // okunacak değerlerin sayısı
   );
```

Parametreler

buffer_index

[in] Indis tamponu.

&data[]

[in] OpenCL tamponunun yazılacağı dizi.

cl_buffer_offset

[in] OpenCL tamponunda okuma işleminin başlayacağı bayt bazındaki konum değeri.

data_array_offset

[in] OpenCL tamponunun değerlerini yazmak için, ilk dizi elemanının indisi.

data_array_count

[in] Okunacak değerlerin sayısı.

Dönüş Değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

BufferWrite

Bir diziyi belirtilen indis üzerindeki tampona yazar.

```
template<typename T>
bool  BufferWrite(
   const int   buffer_index,          // tampon indisi
   T           &data[],               // değer dizisi
   const uint  cl_buffer_offset,      // OpenCL tamponunun başlangıç konumu (bayt cins
   const uint  data_array_offset,     // yazılacak ilk dizi elemanının indisi
   const uint  data_array_count       // yazılacak dizi elemanlarının sayısı
   );
```

Parametreler

buffer_index

[in] Indis tamponu.

&data[]

[in] OpenCL tamponuna yazılacak verilerden oluşan dizi.

cl_buffer_offset

[in] OpenCL tamponunda yazma işleminin başlayacağı bayt bazındaki konum değeri.

data_array_offset

[in] OpenCL tamponuna değerleri yazmak için, ilk dizi elemanının indisi.

data_array_count

[in] Yazılacak değerlerin sayısı.

Dönüş Değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Execute

Belirtilen indis üzerindeki OpenCL programını çalıştırır.

```
bool  Execute(
   const int   kernel_index,           // kernel indisi
   const int   work_dim,               // görev uzayının boyut sayısı
   const uint  &work_offset[],         // görev uzayındaki başlangıç konumu
   const uint  &work_size[]            // görevlerin toplam sayısı
   );
```

Belirtilen indise ve yerel grup üzerindeki görev sayısına göre OpenCL kernelini çalıştırır.

```
bool  Execute(
   const int   kernel_index,           // kernel indisi
   const int   work_dim,               // görev uzayının boyut sayısı
   const uint  &work_offset[],         // görev uzayındaki başlangıç konumu
   const uint  &work_size[],           // toplam görev sayısı
   const uint  &local_work_size[]      // yerel gruptaki görevlerin sayısı
   );
```

Parametreler

kernel_index

[in] Kernel nesnesinin indisi.

work_dim

[in] Görev alanının boyutu.

&work_offset[]

[in][out] Görev uzayındaki başlangıç konumu. Referans ile geçirilir.

&work_size[]

[in][out] Görev alt kümesinin boyutu. Referans ile geçirilir.

&local_work_size[]

[in][out] Gruptaki yerel görev alt kümesinin boyutu. Referans ile geçirilir.

Dönüş Değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

GetContext

OpenCL bağlamının tanıtıcı değerine dönüş yapar.

```
int  GetContext();
```

Dönüş Değeri

OpenCL bağlamının tanıtıcı değeri.

GetKernel

Belirtilen indis üzerindeki kernel nesnesinin tanıtıcı değerine dönüş yapar.

```
int  GetKernel(
   const int  kernel_index      // kernel indisi
   );
```

Parametreler

kernel_index

[in] Kernel nesnesinin indisi.

Dönüş Değeri

Kernel nesnesinin tanıtıcı değeri.

GetKernelName

Belirtilen indis üzerindeki kernel nesnesinin ismine dönüş yapar.

```
string  GetKernelName(
   const int  kernel_index      // kernel indisi
   );
```

Parametreler

kernel_index

[in] Kernel nesnesinin indisi.

Dönüş Değeri

Kernel nesnesinin ismi.

GetProgram

OpenCL programının tanıtıcı değerine dönüş yapar.

```
int  GetProgram();
```

Dönüş Değeri

OpenCL programının tanıtıcı değeri.

Initialize

Belirtilen OpenCL programını başlatır.

```
bool  Initialize(
   const string  program,           // OpenCL programının tanıtıcı değeri
   const bool    show_log=true      // kayıt tut
   );
```

Parametreler

program

[in] OpenCL programının tanıtıcı değeri.

show_log=true

[in] Mesaj kaydını etkinleştir.

Dönüş Değeri

Başarılı ise 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

KernelCreate

Belirtilen indis üzerindeki OpenCL programına bir giriş noktası ekler

```
bool  KernelCreate(
   const int     kernel_index,     // kernel indisi
   const string  kernel_name       // kernelin ismi
   );
```

Parametreler

kernel_index

[in] Kernel nesnesinin indisi.

kernel_name

[in] Kernel nesnesinin ismi.

Dönüş Değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

KernelFree

Belirtilen indis üzerindeki OpenCL start fonksiyonunu siler.

```
bool  KernelFree(
   const int  kernel_index      // kernel indisi
   );
```

Parametreler

kernel_index

[in] Kernel nesnesinin indisi.

Dönüş Değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

SetArgument

Belirtilen indis üzerindeki OpenCL fonksiyonu için bir parametre ayarlar.

```
template<typename T>
bool  SetArgument(
   const int  kernel_index,     // kernel indisi
   const int  arg_index,        // fonksiyon argümanının indisi
   T          value             // kaynak kodu
   );
```

Parametreler

kernel_index

[in] Kernel nesnesinin indisi.

arg_index

[in] Fonksiyon argümanının indisi.

value

[in] Fonksiyon argümanının değeri.

Dönüş Değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

SetArgumentBuffer

Bir OpenCL tamponunu, belirtilen indis üzerindeki OpenCL fonksiyonunun parametresi olarak ayarlar

```
bool  SetArgumentBuffer(
   const int  kernel_index,     // kernel indisi
   const int  arg_index,        // fonksiyon argümanının indisi
   const int  buffer_index      // tampon indisi
   );
```

Parametreler

kernel_index

[in] Kernel nesnesinin indisi.

arg_index

[in] Fonksiyon argümanının indisi.

buffer_index

[in] Indis tamponu.

Dönüş Değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

SetArgumentLocalMemory

Belirtilen indis üzerindeki OpenCL fonksiyonu için yerel bellekte bir parametre ayarlar.

```
bool  SetArgumentLocalMemory(
   const int  kernel_index,          // kernel indisi
   const int  arg_index,             // fonksiyon argümanının indisi
   const int  local_memory_size      // yerel bellek boyutu
   );
```

Parametreler

kernel_index

[in] Kernel nesnesinin indisi.

arg_index

[in] Fonksiyon argümanının indisi.

local_memory_size

[in] Yerel bellek boyutu.

Dönüş Değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

SetBuffersCount

Tampon sayısını ayarlar.

```
bool  SetBuffersCount(
   const int  total_buffers      // Tamponların sayısı
   );
```

Parametreler

total_buffers

[in] Tamponların toplam sayısı.

Dönüş Değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

SetKernelsCount

Kernel nesnelerinin sayısını ayarlar.

```
bool  SetKernelsCount(
   const int  total_kernels      // kernellerin sayısı
   );
```

Parametreler

total_kernels

[in] Kernellerin toplam sayısı.

Dönüş Değeri

Başarı durumunda 'true' değerine, aksi durumda 'false' değerine dönüş yapar.

Shutdown

OpenCL programını kaldırır.

```
void  Shutdown();
```

Dönüş Değeri

Dönüş değeri yok.

SupportDouble

Kayan noktalı veri tiplerinin desteklenip desteklenmediğini kontrol eder.

```
bool  SupportDouble();
```

Dönüş Değeri

Kayan noktalı veri tipleri destekleniyorsa "true", aksi durumda "false" değerine dönüş yapar.

CObject Temel Sınıfı

CObject, MQL5 Standart kütüphanesinin inşa edilmesi için kullanılan temel sınıftır.

Açıklama

CObject, soyundan gelen tüm sınıfların bağlantılı bir listenin parçası olmasını sağlar. Ayrıca, soyundan gelen sınıflara, daha ileri uygulamalar için bazı sanal yöntemler tanımlar.

Bildirim

```
class CObject
```

Başlık

```
#include <Object.mqh>
```

Kalıtım hiyerarşisi

CObject

İlk nesil

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

Sınıf Yöntemleri

Özellikler	
Prev	Bir önceki bileşenin değerini alır
Prev	Bir önceki bileşenin değerini ayarlar
Next	Bir sonraki bileşenin değerini alır
Next	Bir sonraki bileşenin değerini ayarlar
Karşılaştırma yöntemleri	
virtual Compare	Başka bir nesne ile karşılaştırma işleminin sonucunu verir
Girdi/çıktı	
virtual Save	Nesneyi dosyaya yazar
virtual Load	Nesneyi dosyadan okur
virtual Type	Nesnenin tipine dönüş yapar

Prev

Listedeki bir önceki elemanın işaretçisini alır.

```
CObject*  Prev()
```

Dönüş Değeri

Listedeki bir önceki elemanın işaretçisi. Listedeki ilk eleman için NULL dönüşü yapar.

Örnek:

```
//--- CObject::Prev() için bir örnek
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete object_first;
      return;
     }
   //--- etkileşimi ayarla
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- bir önceki nesneyi kullan
   CObject *object=object_second.Prev();
   //--- nesneleri sil
   delete object_first;
   delete object_second;
  }
```

Prev

Listedeki bir önceki elemanın işaretçisini ayarlar.

```
void  Prev(
   CObject*  object      // Bir önceki liste elemanının işaretçisi
   )
```

Parametreler

object

[in] Bir önceki liste elemanının işaretçisi için yeni değer.

Örnek:

```
//--- eCObject::Prev(CObject*) için bir örnek
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete object_first;
      return;
     }
   //--- etkileşimi ayarla
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- nesneleri kullan
   //--- ...
   //--- nesneleri sil
   delete object_first;
   delete object_second;
  }
```

Next

Listedeki bir sonraki elemanın işaretçisini alır.

```
CObject*  Next()
```

Dönüş Değeri

Listedeki bir sonraki elemanın işaretçisi. Listedeki son elemana gelindiğinde ise NULL dönüşü yapar.

Örnek:

```
//--- CObject::Next() için bir örnek
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete object_first;
      return;
     }
   //--- etkileşimi ayarla
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- sonraki nesneyi kullan
   CObject *object=object_first.Next();
   //--- nesneleri sil
   delete object_first;
   delete object_second;
  }
```

Next

Bir sonraki liste elemanının işaretçisi için yeni değer ayarlar.

```
void  Next(
   CObject*  object      // Bir sonraki liste elemanının işaretçisi
   )
```

Parametreler

object

[in] Bir sonraki liste elemanının işaretçisi için yeni değer.

Örnek:

```
//--- CObject::Next(CObject*) için bir örnek
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete object_first;
      return;
     }
   //--- etkileşimi ayarla
   object_first.Next(object_second);
   object_second.Prev(object_first);
   //--- nesneleri kullan
   //--- ...
   //--- nesneleri sil
   delete object_first;
   delete object_second;
  }
```

Compare

Listedeki belli bir bileşenin verilerini başka bir bileşenin verisi ile karşılaştırır.

```
virtual int  Compare(
   const CObject*  node,      // Karşılaştırılacak bileşenin işaretçisi
   const int       mode=0     // Karşılaştırma kipi
   ) const
```

Parametreler

node

[in] Karşılaştırılacak liste bileşeninin işaretçisi

mode=0

[in] Karşılaştırma kipi

Dönüş Değeri

Liste elemanlarının eşit olması durumunda 0, liste elemanı karşılaştırma elemanından daha küçükse -1, liste elemanı karşılaştırma elemanından daha büyükse 1.

Not

CObject sınıfındaki Compare () yöntemi her zaman 0 dönüşü yapar ve herhangi bir eylem gerçekleştirmez. Veri karşılaştırması yapmak istiyorsanız türetik sınıfların Compare (...) yöntemini kullanmalısınız. Ayrıca, 'mode' parametresi çok değişkenli karşılaştırmalar yaparken kullanılmalıdır.

Örnek:

```
//--- CObject::Compare(...) için bir örnek
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object_first,*object_second;
   //---
   object_first=new CObject;
   if(object_first==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   object_second=new CObject;
   if(object_second==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete object_first;
      return;
     }
   //--- etkileşimi ayarla
   object_first.Next(object_second);
```

```
   object_second.Prev(object_first);
   //--- nesneleri karşılaştır
   int result=object_first.Compare(object_second);
   //--- nesneleri sil
   delete object_first;
   delete object_second;
  }
```

Save

Listedeki bir veri bileşenini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Not

CObject sınıfındaki Save (int) yöntemi her zaman 'true' dönüşü yapar ve herhangi bir eylem gerçekleştirmez. Bir dosyadan veri yüklemeye çalışıyorsanız, türetik bir sınıfın Save (int) yöntemini kullanmalısınız.

Örnek:

```
//--- CObject::Save(int) için bir örnek
#include <Object.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CObject *object=new CObject;
   //---
   if(object!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- veri nesnesini ayarla
   //--- . . .
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Save(file_handle))
        {
         //--- dosya kaydetme hatası
         printf("Dosya kaydedilemedi: Hata %d!",GetLastError());
         delete object;
         FileClose(file_handle);
         //---
         return;
```

```
         }
       FileClose(file_handle);
      }
   delete object;
  }
```

Load

Listedeki veri bileşenini dosyadan yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Not

CObject sınıfındaki Load (int) yöntemi her zaman 'true' dönüşü yapar ve herhangi bir eylem gerçekleştirmez. Bir dosyadan veri yüklemeye çalışıyorsanız, türetik bir sınıfın Load (int) yöntemini kullanmalısınız.

Örnek:

```
//--- CObject::Load(int) için bir örnek
#include <Object.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CObject *object=new CObject;
   //---
   if(object!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Load(file_handle))
        {
         //--- dosya yükleme hatası
         printf("Dosya yüklenemedi: Hata %d!",GetLastError());
         delete object;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

```
      }
   //--- nesneyi kullan
   //--- . . .
   delete object;
  }
```

Type

Tip tanımlayıcısını alır.

```
virtual int  Type() const
```

Dönüş Değeri

Tip tanımlayıcısı (CObject için 0).

Örnek:

```
//--- CObject::Type() için bir örnek
#include <Object.mqh>
//---
void OnStart()
  {
   CObject *object=new CObject;
   //---
   object=new CObject;
   if(object ==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- nesne tipini al
   int type=object.Type();
   //--- nesneyi sil
   delete object;
  }
```

Veri Yapıları

Bu bölüm, çeşitli veri yapıları (diziler, bağlı listeler, vb.) ile çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için gereken açıklamaları içermektedir.

Veri yapısı sınıflarının kullanımı çeşitli biçimlerdeki veri depoları (kompozit veri yapıları da dahil) oluştururken zaman kazandırır.

Veri setleri için geliştirilmiş MQL5 Standart Kütüphanesi bileşenleri terminalin çalışma dizininde Include\Arrays klasöründe yer alır.

Veri dizileri

Dinamik veri dizileri için geliştirilen sınıfların kullanımı çeşitli biçimlerdeki veri depolarının oluşturulması sırasında zaman kazandıracaktır.

Veri dizileri için geliştirilen MQL5 Standart Kütüphanesi bileşenleri terminalin çalışma dizininde Include\Arrays klasöründe yer alır.

Sınıf	Açıklama
CArray	Dinamik veri dizisi için temel sınıf
CArrayChar	char veya uchar tipli değişkenler için dinamik dizi sınıfı
CArrayShort	short veya ushort tipli değişkenler için dinamik dizi sınıfı
CArrayInt	int veya uint tipli değişkenler için dinamik dizi sınıfı
CArrayLong	long veya ulong tipli değişkenler için dinamik dizi sınıfı
CArrayFloat	float tipli değişkenler için dinamik dizi sınıfı
CArrayDouble	double tipli değişkenler için dinamik dizi sınıfı
CArrayString	string tipli değişkenler için dinamik dizi sınıfı
CArrayObj	Dinamik işaretçi dizisi CObject
CList	CObject sınıfının ve türevlerinin örneklerinden oluşan bir liste ile çalışabilme imkanı sağlar
CTreeNode	CTree ikili ağaç ağları ile çalışabilme imkanı sağlar
CTree	CTreeNode sınıfının ve türevlerinin örneklerinden oluşan ikili ağaç ile çalışabilme olanağı sağlar

CArray

CArray sınıfı, dinamik diziler için bir temel sınıftır.

Açıklama

CArray sınıfı, dinamik diziler için bellek tahsis yönetimi, sıralama ve dosya işlemleri gibi çalışmaların yapılabilmesine olanak sağlar.

Bildirim

```
class CArray : public CObject
```

Başlık

```
#include <Arrays\Array.mqh>
```

Kalıtım hiyerarşisi

CObject

CArray

İlk nesil

CArrayChar, CArrayDouble, CArrayFloat, CArrayInt, CArrayLong, CArrayObj, CArrayShort, CArrayString

Sınıf Yöntemleri

Özellikler	
Step	Dizinin artış büyüklüğünü alır
Step	Dizinin artış büyüklüğünü ayarlar
Total	Dizi elemanlarının sayısını alır
Available	Yeni bellek tahsisi olmadan dizinin mevcut serbest eleman sayısını alır
Max	Yeni bellek tahsisi olmadan dizinin mümkün olan en büyük boyutunu alır
IsSorted	Dizinin belirtilen şekilde sıralanıp sıralanmadığını doğrular
SortMode	Dizinin sıralama versiyonunu alır
Temizleme yöntemleri	
Clear	Belleği boşaltmadan dizi elemanlarının tamamını temizler
Sıralama yöntemleri	
Sort	Belirtilen şekilde diziyi sıralar
Girdi/çıktı	
virtual Save	Veri dizisini dosyaya kaydeder

Özellikler	
virtual Load	Veri dizisini dosyadan alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

Step

Dizinin artış büyüklüğünü alır.

```
int  Step() const
```

Dönüş Değeri

Dizinin artış büyüklüğünü.

Örnek:

```
//--- CArray::Step() için bir örnek
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- artış büyüklüğünü al
   int step=array.Step();
   //--- diziyi kullan
   //--- ...
   //--- diziyi sil
   delete array;
  }
```

Step

Dizinin artış büyüklüğünü ayarlar.

```
bool  Step(
   int  step      // adım
   )
```

Parametreler

step

[in] Dizinin artış büyüklüğü (adım) için yeni değer.

Dönüş Değeri

Başarılı ise 'true', adım değeri sıfır veya daha küçük bir sayıyla ayarlanmaya çalışılmışsa 'false'.

Örnek:

```
//--- CArray::Step(int) için bir örnek
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- artış büyüklüğünü ayarla
   bool result=array.Step(1024);
   //--- diziyi kullan
   //--- ...
   //--- diziyi sil
   delete array;
  }
```

Total

Dizi elemanlarının toplam sayısını alır.

```
int  Total() const;
```

Dönüş Değeri

Dizideki elemanların sayısı.

Örnek:

```
//--- CArray::Total() için bir örnek
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- toplamı kontrol et
   int total=array.Total();
   //--- diziyi kullan
   //--- ...
   //--- diziyi sil
   delete array;
  }
```

Available

Yeni bellek tahsisi olmadan dizideki mevcut serbest eleman sayısını alır.

```
int  Available() const
```

Dönüş Değeri

Yeni bellek tahsisi olmadan dizideki mevcut serbest eleman sayısı.

Örnek:

```
//--- CArray::Available() için bir örnek
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kullanılabilir alanı denetle
   int available=array.Available();
   //--- diziyi kullan
   //--- ...
   //--- diziyi sil
   delete array;
  }
```

Max

Yeni bellek tahsisi olmadan dizinin mümkün olan en büyük boyutunu alır.

```
int  Max() const
```

Dönüş Değeri

Yeni bellek tahsisi olmadan dizinin mümkün olan en büyük boyutunu alır.

Örnek:

```
//--- CArray::Max() için bir örnek
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- maksimum boyutu kontrol et
   int max=array.Max();
   //--- diziyi kullan
   //--- ...
   //--- diziyi sil
   delete array;
  }
```

IsSorted

Dizinin belirtilen şekilde sıralanıp sıralanmadığını doğrular.

```
bool  IsSorted(
   int  mode=0      // Sıralama kipi
   ) const
```

Parametreler

mode=0

[in] Test edilecek sıralama türü.

Dönüş Değeri

Sıralı listenin bayrağı. Liste istenen yolla sıralanmışsa 'true', aksi durumda 'false'.

Not

Dizinin sıralanma bayrağı doğrudan değiştirilemez. Sıralanma bayrağı Sort () yöntemi ile ayarlanır ve InsertSort (...) haricindeki ekleme yöntemlerini sıfırlar.

Örnek:

```
//--- CArray::IsSorted() için bir örnek
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- sıralamayı denetle
   if(array.IsSorted())
     {
      //--- sıralı dizi için yöntemler kullan
      //--- ...
     }
   //--- diziyi sil
   delete array;
  }
```

SortMode

Dizinin sıralanış türünü alır.

```
int  SortMode() const;
```

Dönüş Değeri

Sıralama kipi.

Örnek:

```
//--- CArray::SortMode() için bir örnek
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- sıralama türünü kontrol et
   int sort_mode=array.SortMode();
   //--- diziyi kullan
   //--- ...
   //--- diziyi sil
   delete array;
  }
```

Clear

Belleği boşaltmadan dizi elemanlarının tamamını temizler.

```
void  Clear()
```

Dönüş Değeri

Yok.

Örnek:

```
//--- CArray::Clear() için bir örnek
#include <Arrays\Array.mqh>
//---
void OnStart()
 {
 CArray *array=new CArray;
 //---
 if(array==NULL)
 {
 printf("Nesne oluşturma hatası");
 return;
 }
 //--- diziyi kullan
 //--- ...
 //--- diziyi temizle
 array.Clear();
 //--- diziyi sil
 delete array;
 }
```

Sort

Belirtilen şekilde diziyi sıralar.

```
void  Sort(
   int  mode=0      // Sıralama kipi
   )
```

Parametreler

mode=0

[in] Dizi sıralama kipi.

Dönüş Değeri

Yok.

Not

Dizilerin sıralanması her zaman artan şekilde gerçekleşir. Basit tiplerdeki verilerden oluşan diziler için (CArrayChar, CArrayShort, vb.), "mode" parametresi kullanılmaz. CArrayObj dizisi için, Sort (int) yöntemi ile çok değişkenli sıralama uygulanmalıdır.

Örnek:

```
//--- CArray::Sort(int) için bir örnek
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   CArray *array=new CArray;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- mode 0 ile sıralama
   array.Sort(0);
   //--- diziyi kullan
   //--- ...
   //--- diziyi sil
   delete array;
  }
```

Save

Veri dizisini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CArray::Save(int) için bir örnek
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   int     file_handle;
   CArray *array=new CArray;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- dosya kaydetme hatası
         printf("Dosya kaydedilemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- diziyi sil
   delete array;
  }
```

Load

Belirtilen dosyadan veri dizisi yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CArray::Load(...) için bir örnek
#include <Arrays\Array.mqh>
//---
void OnStart()
  {
   int     file_handle;
   CArray *array=new CArray;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- dosya yükleme hatası
         printf("Dosya yüklenemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- diziyi sil
   delete array;
  }
```

CArrayChar

CArrayChar sınıfı, char veya uchar tipli değişkenler için bir dinamik dizi sınıfıdır.

Açıklama

CArrayChar sınıfı, char veya uchar tipli değişkenlerden oluşan dinamik dizilerle çalışma olanağı sağlar. Sınıf içerisinde; sıralama, sıralanmış dizide arama, eleman silme/ekleme gibi uygulamalara yer verilmiştir. Ayrıca dosya işlemleri için gereken yöntemler de sınıf içerisinde yer almaktadır.

Bildirim

```
class CArrayChar : public CArray
```

Başlık

```
#include <Arrays\ArrayChar.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayChar

Sınıf Yöntemleri

Bellek denetimi	
Reserve	Dizi boyutunun artırılması için bellek tahsis eder
Resize	Dizi için daha küçük bir boyut ayarlar
Shutdown	Belleği boşaltarak diziyi siler
Ekleme yöntemleri	
Add	Dizinin sonuna bir eleman ekler
AddArray	Başka bir diziden alınan elemanları dizinin sonuna ekler
AddArray	Başka bir diziden alınan elemanları dizinin sonuna ekler
Insert	Dizinin belirtilen konumuna eleman ekler
InsertArray	Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler
InsertArray	Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler
AssignArray	Dizinin elemanlarını başka bir diziden kopyalar
AssignArray	Dizinin elemanlarını başka bir diziden kopyalar
Değiştirme yöntemleri	
Update	Dizinin belirtilen konumundaki elemanı değiştirir

Bellek denetimi	
Shift	Dizi içindeki bir elemanı belirtilen konuma kaydırır
Silme yöntemleri	
Delete	Dizinin belirtilen konumundaki elemanı siler
DeleteRange	Dizinin belirtilen konumundaki elemanlar grubunu siler
Erişim yöntemleri	
At	Dizinin belirtilen konumundaki elemanı alır
Karşılaştırma yöntemleri	
CompareArray	Diziyi başka bir diziyle karşılaştırır
CompareArray	Diziyi başka bir diziyle karşılaştırır
Sıralı dizi yöntemleri	
InsertSort	Sıralı diziye eleman ekler
Search	Belirtilen örneğin sıralı dizideki eşitini arar
SearchGreat	Sıralı dizi içinde, belirtilen örnekten daha büyük olan bir eleman arar
SearchLess	Sıralı dizi içinde, belirtilen örnekten daha küçük olan bir eleman arar
SearchGreatOrEqual	Sıralı dizi içinde, belirtilen örnekten daha büyük veya eşit olan bir eleman arar
SearchLessOrEqual	Sıralı dizi içinde, belirtilen örnekten daha küçük veya eşit olan bir eleman arar
SearchFirst	Sıralı dizi içinde, belirtilen kriteri sağlayan ilk elemanı arar
SearchLast	Sıralı dizi içinde, belirtilen kriteri sağlayan son elemanı arar
SearchLinear	Belirtilen örneğin dizi içindeki eşitini arar
Girdi/çıktı	
virtual Save	Veri dizisini dosyaya kaydeder
virtual Load	Veri dizisini dosyadan alır
virtual Type	Dizinin tip tanımlayıcısını alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Reserve

Dizi boyutunun artırılması için bellek tahsis eder.

```
bool  Reserve(
   int  size      // eleman sayısı
   )
```

Parametreler

size

[in] Eklenecek elemanların sayısı.

Dönüş Değeri

Başarılı ise 'true', eleman sayısı sıfır veya daha küçük bir değerle ayarlanmaya çalışılmışsa 'false'.

Not

Bellek bölünmesini azaltmak için, daha önce Step (int) yöntemi ile ayarlanmış olan büyüklük değerini artırın (varsayılan değer 16).

Örnek:

```
//--- CArrayChar::Reserve(int) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- bellek tahsis et
   if(!array.Reserve(1024))
     {
      printf("Tahsis hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Resize

Dizi için daha küçük bir boyut ayarlar.

```
bool  Resize(
   int  size      // Boyut
   )
```

Parametreler

size

[in] Yeni dizi boyutu.

Dönüş Değeri

Başarılı ise 'true', boyut değeri sıfır veya daha küçük bir sayıyla ayarlanmaya çalışılmışsa 'false'.

Not

Dizi boyutunun değiştirilmesi belleğin etkin kullanımını sağlar. sağ taraftaki gereksiz elemanlar kaybedilir. Bellek bölünmesini azaltmak için, daha önce Step (int) yöntemi ile ayarlanmış olan büyüklük değerini azaltın (varsayılan değer 16).

Örnek:

```
//--- CArrayChar::Resize(int) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi yeniden boyutlandır
   if(!array.Resize(10))
     {
      printf("Boyutlandırma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Shutdown

Belleği boşaltarak diziyi siler.

```
bool  Shutdown()
```

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayChar::Shutdown() örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi kapat
   if(!array.Shutdown())
     {
      printf("Kapatma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Add

Dizinin sonuna bir eleman ekler.

```
bool  Add(
   char  element      // eklenecek eleman
   )
```

Parametreler

element

[in] Diziye eklenecek elemanın değeri.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayChar::Add(char) için bir örnek,i
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Eleman ekleme hatası");
         delete array;
         return;
        }
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AddArray

Başka bir diziden alınan elemanları dizinin sonuna ekler.

```
bool  AddArray(
   const char&  src[]      // kaynak dizi
   )
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayChar::AddArray(const char &[]) için bir örnek
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- yeni dizi ekle
   if(!array.AddArray(src))
     {
      printf("Dizi ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AddArray

Başka bir diziden alınan elemanları dizinin sonuna ekler.

```
bool  AddArray(
   const CArrayChar*  src      // kaynağın işaretçisi
   )
```

Parametreler

src

[in] Bir CArrayChar sınıf örneğinin (eklenecek kaynak elemanlar) işaretçisi.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayChar::AddArray(const CArrayChar*) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- yeni dizi ekle
   if(!array.AddArray(src))
     {
      printf("Dizi ekleme hatası");
      delete src;
      delete array;
      return;
     }
   //--- kaynak diziyi sil
   delete src;
```

```
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Insert

Dizinin belirtilen konumuna eleman ekler.

```
bool  Insert(
   char  element,     // eklenecek eleman
   int   pos          // konum
   )
```

Parametreler

element

[in] Diziye eklenecek elemanın değeri

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayChar::Insert(char,int) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- eleman ekle
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Ekleme hatası");
         delete array;
         return;
        }
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

InsertArray

Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler.

```
bool  InsertArray(
   const char&  src[],     // kaynak dizi
   int           pos       // konum
   )
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', elemanlar eklenemezse 'false'.

Örnek:

```
//--- CArrayChar::InsertArray(const char &[],int) örneği
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ekle
   if(!array.InsertArray(src,0))
     {
      printf("dizi ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

InsertArray

Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler.

```
bool  InsertArray(
   CArrayChar*  src,      // kaynağın işaretçisi
   int           pos      // konum
   )
```

Parametreler

src

[in] Bir CArrayChar sınıf örneğinin (eklenecek kaynak elemanlar) işaretçisi.

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', elemanlar eklenemezse 'false'.

Örnek:

```
//--- CArrayChar::InsertArray(const CArrayChar*,int) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ekle
   if(!array.InsertArray(src,0))
     {
      printf("dizi ekleme hatası");
      delete src;
      delete array;
```

```
      return;
     }
   //--- kaynak diziyi sil
   delete src;
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AssignArray

Dizinin elemanlarını başka bir diziden kopyalar

```
bool  AssignArray(
   const char&  src[]      // kaynak dizi
   )
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

Dönüş Değeri

Başarılı ise 'true', elemanlar kopyalanamazsa 'false'.

Örnek:

```
//--- CArrayChar::AssignArray(const char &[]) örneği
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ata
   if(!array.AssignArray(src))
     {
      printf("Dizi atama hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AssignArray

Dizinin elemanlarını başka bir diziden kopyalar

```
bool  AssignArray(
   const CArrayChar*  src      // kaynağın işaretçisi
   )
```

Parametreler

src

[in] Bir CArrayChar sınıf örneğinin (kopyalanacak kaynak elemanlar) işaretçisi.

Dönüş Değeri

Başarılı ise 'true', elemanlar kopyalanamazsa 'false'.

Örnek:

```
//--- CArrayChar::AssignArray(const CArrayChar*) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayChar *src  =new CArrayChar;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ata
   if(!array.AssignArray(src))
     {
      printf("Dizi atama hatası");
      delete src;
      delete array;
      return;
     }
   //--- diziler aynı
   //--- kaynak diziyi sil
```

```
   delete src;
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Update

Dizinin belirtilen konumundaki elemanı değiştirir.

```
bool  Update(
   int   pos,          // konum
   char  element       // değer
   )
```

Parametreler

pos

[in] Değiştirilecek elemanın dizideki konumu

element

[in] Elemanın yeni değeri

Dönüş Değeri

Başarılı ise 'true', eleman değiştirilemezse 'false'.

Örnek:

```
//--- CArrayChar::Update(int,char) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı güncelle
   if(!array.Update(0,'A'))
     {
      printf("Güncelleme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Shift

Dizi içindeki bir elemanı belirtilen konuma kaydırır.

```
bool  Shift(
   int  pos,        // konum
   int  shift       // kaydırma değeri
   )
```

Parametreler

pos

[in] Kaydırılacak dizi elemanının konumu

shift

[in] Kaydırma değeri (pozitif veya negatif).

Dönüş Değeri

Başarılı ise 'true', eleman kaydırılamadıysa 'false'.

Örnek:

```
//--- CArrayChar::Shift(int,int) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı kaydır
   if(!array.Shift(10,-5))
     {
      printf("Kaydırma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Delete

Dizinin belirtilen konumundaki elemanı siler.

```
bool  Delete(
   int  pos      // konum
   )
```

Parametreler

pos

[in] Silinecek elemanın konumu.

Dönüş Değeri

Başarılı ise 'true', eleman silinemezse 'false'.

Örnek:

```
//--- CArrayChar::Delete(int) için bir örnek
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı sil
   if(!array.Delete(0))
     {
      printf("Silme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

DeleteRange

Dizinin belirtilen konumundaki elemanlar grubunu siler.

```
bool  DeleteRange(
   int  from,     // ilk elemanın konumu
   int  to        // son elemanın konumu
   )
```

Parametreler

from

[in] Silinecek ilk elemanın konumu.

to

[in] Silinecek son elemanın konumu.

Dönüş Değeri

Başarılı ise 'true', elemanlar silinemezse 'false'.

Örnek:

```
//--- CArrayChar::DeleteRange(int,int) için bir örnek
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanları sil
   if(!array.DeleteRange(0,10))
     {
      printf("Silme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

At

Dizinin belirtilen konumundaki elemanı alır.

```
char  At(
   int  pos      // konum
   ) const
```

Parametreler

pos

[in] Seçilen elemanın dizi içindeki konumu.

Dönüş Değeri

Başarı durumunda elemanın değerine, olmayan bir konumdaki bir elemanın istenmesi durumunda ise CHAR_MAX değerine dönüş yapar (son hata değeri ERR_OUT_OF_RANGE).

Not

CHAR_MAX değeri geçerli bir dizi elemanı olabileceği için her zaman son hata kodunun istenmesi gerekir.

Örnek:

```
//--- CArrayChar::At(int) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      char result=array.At(i);
      if(result==CHAR_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- dizi okuma hatası
         printf("Eleman alınamadı, hata");
         delete array;
         return;
        }
      //--- elemanı kullan
      //--- . . .
     }
```

```
   //--- diziyi sil
   delete array;
  }
```

CompareArray

Diziyi başka bir diziyle karşılaştırır.

```
bool  CompareArray(
   const char&  src[]      // kaynak dizi
   ) const
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

Dönüş Değeri

Diziler eşit ise 'true', aksi durumda 'false'

Örnek:

```
//--- CArrayChar::CompareArray(const char &[]) örneği
#include <Arrays\ArrayChar.mqh>
//---
char src[];
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ile karşılaştır
   int result=array.CompareArray(src);
   //--- diziyi sil
   delete array;
  }
```

CompareArray

Diziyi başka bir diziyle karşılaştırır.

```
bool  CompareArray(
   const CArrayChar*  src      // kaynak dizi işaretçisi
   ) const
```

Parametreler

src

[in] Bir CArrayChar sınıf örneğinin (karşılaştırılacak elemanlar) işaretçisi.

Dönüş Değeri

Diziler eşit ise 'true', aksi durumda 'false'

Örnek:

```
//--- CArrayChar::CompareArray(const CArrayChar*) için bir örnek
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayChar *src=new CArrayChar;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ile karşılaştır
   int result=array.CompareArray(src);
   //--- dizileri sil
   delete src;
   delete array;
  }
```

InsertSort

Sıralı diziye eleman ekler.

```
bool  InsertSort(
   char  element      // eklenecek eleman
   )
```

Parametreler

element

[in] Sıralı diziye eklenecek elemanın değeri

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayChar::InsertSort(char) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ekle
   if(!array.InsertSort('A'))
     {
      printf("Ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Search

Belirtilen örneğin sıralı dizideki eşitini arar.

```
int  Search(
   char  element      // Örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayChar::Search(char) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.Search('A')!=-1) printf("Eleman bulundu");
   else                      printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchGreat

Sıralı dizi içinde, belirtilen örnekten daha büyük olan bir eleman arar.

```
int  SearchGreat(
   char  element      // Örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayChar::SearchGreat(char) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchGreat('A')!=-1) printf("Eleman bulundu");
   else                           printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLess

Sıralı dizi içinde, belirtilen örnekten daha küçük olan bir eleman arar.

```
int  SearchLess(
   char  element      // Örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayChar::SearchLess(char) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLess('A')!=-1) printf("Eleman bulundu");
   else                          printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchGreatOrEqual

Sıralı dizi içinde, belirtilen örnekten daha büyük veya eşit olan bir eleman arar

```
int  SearchGreatOrEqual(
   char  element      // Örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayChar::SearchGreatOrEqual(char) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchGreatOrEqual('A')!=-1) printf("Eleman bulundu");
   else                                  printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLessOrEqual

Sıralı dizi içinde, belirtilen örnekten daha küçük veya eşit olan bir eleman arar.

```
int  SearchLessOrEqual(
   char  element      // Örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayChar::SearchLessOrEqual(char) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLessOrEqual('A')!=-1) printf("Eleman bulundu");
   else                                 printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchFirst

Sıralı dizi içinde, belirtilen kriteri sağlayan ilk elemanı arar.

```
int  SearchFirst(
   char  element      // Örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayChar::SearchFirst(char) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchFirst('A')!=-1) printf("Eleman bulundu");
   else                           printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLast

Sıralı dizi içinde, belirtilen kriteri sağlayan son elemanı arar.

```
int  SearchLast(
   char  element      // Örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayChar::SearchLast(char) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLast('A')!=-1) printf("Eleman bulundu");
   else                          printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLinear

Belirtilen örneğin dizi içindeki eşitini arar.

```
int  SearchLinear(
   char  element      // Örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Not

Yöntem, sıralanmamış diziler için lineer arama (veya ardışık arama) algoritmasını kullanır.

Örnek:

```
//--- CArrayChar::SearchLinear(char) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- eleman ara
   if(array.SearchLinear('A')!=-1) printf("Eleman bulundu");
   else                            printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

Save

Veri dizisini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayChar::Save(int) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayChar *array=new CArrayChar;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- dosya kaydetme hatası
         printf("Dosya kaydedilemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- diziyi sil
   delete array;
  }
```

Load

Belirtilen dosyadan veri dizisi yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayChar::Load(int) örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayChar *array=new CArrayChar;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- dosya yükleme hatası
         printf("Dosya yüklenemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- dizi elemanlarını kullan
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = '%c'",i,array.At(i));
```

```
      }
   //--- diziyi sil
   delete array;
  }
```

Type

Dizinin tip tanımlayıcısını alır.

```
virtual int  Type() const
```

Dönüş Değeri

Dizinin tip tanımlayıcısı (CArrayChar için 77).

Örnek:

```
//--- CArrayChar::Type() örneği
#include <Arrays\ArrayChar.mqh>
//---
void OnStart()
  {
   CArrayChar *array=new CArrayChar;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dizi tipini al
   int type=array.Type();
   //--- diziyi sil
   delete array;
  }
```

CArrayShort

CArrayShort sınıfı, short veya ushort tipli değişkenler için bir dinamik dizi sınıfıdır.

Açıklama

CArrayShort sınıfı, short ve ushort tipli değişkenlerden oluşan dinamik dizilerle çalışma olanağı sağlar. Sınıf içerisinde; sıralama, sıralanmış dizide arama, eleman silme/ekleme gibi uygulamalara yer verilmiştir. Ayrıca dosya işlemleri için gereken yöntemler de sınıf içerisinde yer almaktadır.

Bildirim

```
class CArrayShort : public CArray
```

Başlık

```
#include <Arrays\ArrayShort.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayShort

Sınıf Yöntemleri

Bellek denetimi	
Reserve	Dizi boyutunun artırılması için bellek tahsis eder
Resize	Dizi için daha küçük bir boyut ayarlar
Shutdown	Belleği boşaltarak diziyi siler
Ekleme yöntemleri	
Add	Dizinin sonuna bir eleman ekler
AddArray	Başka bir diziden alınan elemanları dizinin sonuna ekler
AddArray	Başka bir diziden alınan elemanları dizinin sonuna ekler
Insert	Dizinin belirtilen konumuna eleman ekler
InsertArray	Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler
InsertArray	Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler
AssignArray	Dizinin elemanlarını başka bir diziden kopyalar
AssignArray	Dizinin elemanlarını başka bir diziden kopyalar
Güncelleme yöntemleri	
Update	Dizinin belirtilen konumundaki elemanı değiştirir

Bellek denetimi	
Shift	Dizi içindeki bir elemanı belirtilen konuma kaydırır
Silme yöntemleri	
Delete	Dizinin belirtilen konumundaki elemanı siler
DeleteRange	Dizinin belirtilen konumundaki elemanlar grubunu siler
Erişim yöntemleri	
At	Dizinin belirtilen konumundaki elemanı alır
Karşılaştırma yöntemleri	
CompareArray	Diziyi başka bir diziyle karşılaştırır
CompareArray	Diziyi başka bir diziyle karşılaştırır
Sıralı dizi işlemleri	
InsertSort	Sıralı diziye eleman ekler
Search	Sıralı dizi içinde, belirtilen örneğe eşit olan bir eleman arar
SearchGreat	Sıralı dizi içinde, belirtilen örnekten daha büyük olan bir eleman arar
SearchLess	Sıralı dizi içinde, belirtilen örnekten daha küçük olan bir eleman arar
SearchGreatOrEqual	Sıralı dizi içinde, belirtilen örnekten daha büyük veya eşit olan bir eleman arar
SearchLessOrEqual	Sıralı dizi içinde, belirtilen örnekten daha küçük veya eşit olan bir eleman arar
SearchFirst	Sıralı dizi içinde, belirtilen kriteri sağlayan ilk elemanı arar
SearchLast	Sıralı dizi içinde, belirtilen kriteri sağlayan son elemanı arar
SearchLinear	Belirtilen örneğin dizi içindeki eşitini arar
Girdi/çıktı	
virtual Save	Veri dizisini dosyaya kaydeder
virtual Load	Veri dizisini dosyadan alır
virtual Type	Dizinin tip tanımlayıcısını alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Reserve

Dizi boyutunun artırılması için bellek tahsis eder.

```
bool  Reserve(
   int  size      // eleman sayısı
   )
```

Parametreler

size

[in] Eklenecek elemanların sayısı.

Dönüş Değeri

Başarılı ise 'true', eleman sayısı sıfır veya daha küçük bir değerle ayarlanmaya çalışılmışsa 'false'.

Not

Bellek bölünmesini azaltmak için, daha önce Step (int) yöntemi ile ayarlanmış olan büyüklük değerini artırın (varsayılan değer 16).

Örnek:

```
//--- CArrayShort::Reserve(int) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- bellek tahsis et
   if(!array.Reserve(1024))
     {
      printf("Tahsis hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Resize

Dizi için daha küçük bir boyut ayarlar.

```
bool  Resize(
   int  size      // Boyut
   )
```

Parametreler

size

[in] Yeni dizi boyutu.

Dönüş Değeri

Başarılı ise 'true', boyut değeri sıfır veya daha küçük bir sayıyla ayarlanmaya çalışılmışsa 'false'.

Not

Dizi boyutunun değiştirilmesi belleğin etkin kullanımını sağlar. sağ taraftaki gereksiz elemanlar kaybedilir. Bellek bölünmesini azaltmak için, daha önce Step (int) yöntemi ile ayarlanmış olan büyüklük değerini azaltın (varsayılan değer 16).

Örnek:

```
//--- CArrayShort::Resize(int) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi yeniden boyutlandır
   if(!array.Resize(10))
     {
      printf("Boyutlandırma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Shutdown

Belleği boşaltarak diziyi siler.

```
bool  Shutdown()
```

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayShort::Shutdown() için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi kapat
   if(!array.Shutdown())
     {
      printf("Kapatma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Add

Dizinin sonuna bir eleman ekler.

```
bool  Add(
   short  element      // eklenecek eleman
   )
```

Parametreler

element

[in] Diziye eklenecek elemanın değeri.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayShort::Add(short) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Eleman ekleme hatası");
         delete array;
         return;
        }
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AddArray

Başka bir diziden alınan elemanları dizinin sonuna ekler.

```
bool  AddArray(
   const short&  src[]      // kaynak dizi
   )
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayShort::AddArray(const short &[]) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- yeni dizi ekle
   if(!array.AddArray(src))
     {
      printf("Dizi ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AddArray

Başka bir diziden alınan elemanları dizinin sonuna ekler.

```
bool  AddArray(
   const CArrayShort*  src      // kaynağın işaretçisi
   )
```

Parametreler

src

[in] CArrayShort sınıfının bir örneğinin işaretçisi (eklenecek kaynak elemanlar).

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayShort::AddArray(const CArrayShort*) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- yeni dizi ekle
   if(!array.AddArray(src))
     {
      printf("Dizi ekleme hatası");
      delete src;
      delete array;
      return;
     }
   //--- kaynak diziyi sil
   delete src;
```

```
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Insert

Dizinin belirtilen konumuna eleman ekler.

```
bool  Insert(
   short  element,      // eklenecek eleman
   int    pos           // konum
   )
```

Parametreler

element

[in] Diziye eklenecek elemanın değeri

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayShort::Insert(short,int) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- eleman ekle
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Ekleme hatası");
         delete array;
         return;
        }
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

InsertArray

Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler.

```
bool  InsertArray(
   const short&  src[],     // kaynak dizi
   int           pos        // konum
   )
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', elemanlar eklenemezse 'false'.

Örnek:

```
//--- CArrayShort::InsertArray(const short &[],int) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ekle
   if(!array.InsertArray(src,0))
     {
      printf("dizi ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

InsertArray

Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler.

```
bool  InsertArray(
   CArrayShort*  src,      // kaynağın işaretçisi
   int            pos      // konum
   )
```

Parametreler

src

[in] CArrayShort sınıf örneğinin (eklenecek kaynak elemanlar) işaretçisi.

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', elemanlar eklenemezse 'false'.

Örnek:

```
//--- CArrayShort::InsertArray(const CArrayShort*,int) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ekle
   if(!array.InsertArray(src,0))
     {
      printf("dizi ekleme hatası");
      delete src;
      delete array;
```

```
      return;
     }
   //--- kaynak diziyi sil
   delete src;
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AssignArray

Dizinin elemanlarını başka bir diziden kopyalar

```
bool  AssignArray(
   const short&  src[]      // kaynak dizi
   )
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

Dönüş Değeri

Başarılı ise 'true', elemanlar kopyalanamazsa 'false'.

Örnek:

```
//--- CArrayShort::AssignArray(const short &[]) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ata
   if(!array.AssignArray(src))
     {
      printf("Dizi atama hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AssignArray

Dizinin elemanlarını başka bir diziden kopyalar

```
bool  AssignArray(
   const CArrayShort*  src      // kaynağın işaretçisi
   )
```

Parametreler

src

[in] CArrayShort sınıf örneğinin (kopyalanacak kaynak elemanlar) işaretçisi.

Dönüş Değeri

Başarılı ise 'true', elemanlar kopyalanamazsa 'false'.

Örnek:

```
//--- CArrayShort::AssignArray(const CArrayShort*) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayShort *src  =new CArrayShort;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ata
   if(!array.AssignArray(src))
     {
      printf("Dizi atama hatası");
      delete src;
      delete array;
      return;
     }
   //--- diziler aynı
   //--- kaynak diziyi sil
```

```
   delete src;
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Update

Dizinin belirtilen konumundaki elemanı değiştirir.

```
bool  Update(
   int    pos,          // konum
   short  element       // değer
   )
```

Parametreler

pos

[in] Değiştirilecek elemanın dizideki konumu

element

[in] Elemanın yeni değeri

Dönüş Değeri

Başarılı ise 'true', eleman değiştirilemezse 'false'.

Örnek:

```
//--- CArrayShort::Update(int,short) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı güncelle
   if(!array.Update(0,100))
     {
      printf("Güncelleme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Shift

Dizi içindeki bir elemanı belirtilen konuma kaydırır.

```
bool  Shift(
   int  pos,        // konumlar
   int  shift       // Kaydırma değeri
   )
```

Parametreler

pos

[in] Kaydırılacak dizi elemanının konumu

shift

[in] Kaydırma değeri (pozitif veya negatif).

Dönüş Değeri

Başarılı ise 'true', eleman kaydırılamadıysa 'false'.

Örnek:

```
//--- CArrayShort::Shift(int,int) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı kaydır
   if(!array.Shift(10,-5))
     {
      printf("Kaydırma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Delete

Dizinin belirtilen konumundaki elemanı siler.

```
bool  Delete(
   int  pos      // konum
   )
```

Parametreler

pos

[in] Silinecek elemanın konumu.

Dönüş Değeri

Başarılı ise 'true', eleman silinemezse 'false'.

Örnek:

```
//--- CArrayShort::Delete(int) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı sil
   if(!array.Delete(0))
     {
      printf("Silme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

DeleteRange

Dizinin belirtilen konumundaki elemanlar grubunu siler.

```
bool  DeleteRange(
   int  from,     // ilk elemanın konumu
   int  to        // son elemanın konumu
   )
```

Parametreler

from

[in] Silinecek ilk elemanın konumu.

to

[in] Silinecek son elemanın konumu.

Dönüş Değeri

Başarılı ise 'true', elemanlar silinemezse 'false'.

Örnek:

```
//--- CArrayShort::DeleteRange(int,int) çin bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanları sil
   if(!array.DeleteRange(0,10))
     {
      printf("Silme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

At

Dizinin belirtilen konumundaki elemanı alır.

```
short  At(
   int  pos      // konum
   ) const
```

Parametreler

pos

[in] Seçilen elemanın dizi içindeki konumu.

Dönüş Değeri

Başarı durumunda elemanın değerine, olmayan bir konumdaki bir elemanın istenmesi durumunda ise SHORT_MAX değerine dönüş yapar (son hata değeri ERR_OUT_OF_RANGE).

Not

SHORT_MAX değeri geçerli bir dizi elemanı olabileceği için her zaman son hata kodunun istenmesi gerekir.

Örnek:

```
//--- CArrayShort::At(int) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      short result=array.At(i);
      if(result==SHORT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- dizi okuma hatası
         printf("Eleman alınamadı, hata");
         delete array;
         return;
        }
      //--- elemanı kullan
      //--- . . .
     }
```

```
   //--- diziyi sil
   delete array;
  }
```

CompareArray

Diziyi başka bir diziyle karşılaştırır.

```
bool  CompareArray(
   const short&  src[]      // kaynak dizi
   ) const
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

Dönüş Değeri

Diziler eşit ise 'true', aksi durumda 'false'

Örnek:

```
//--- CArrayShort::CompareArray(const short &[]) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
short src[];
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ile karşılaştır
   int result=array.CompareArray(src);
   //--- diziyi sil
   delete array;
  }
```

CompareArray

Diziyi başka bir diziyle karşılaştırır.

```
bool  CompareArray(
   const CArrayShort*  src      // kaynağın işaretçisi
   ) const
```

Parametreler

src

[in] CArrayShort sınıf örneğinin (karşılaştırılacak elemanlar) işaretçisi.

Dönüş Değeri

Diziler eşit ise 'true', aksi durumda 'false'

Örnek:

```
//--- CArrayShort::CompareArray(const CArrayShort*) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayShort *src=new CArrayShort;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ile karşılaştır
   int result=array.CompareArray(src);
   //--- dizileri sil
   delete src;
   delete array;
  }
```

InsertSort

Sıralı diziye eleman ekler.

```
bool  InsertSort(
   short  element      // eklenecek eleman
   )
```

Parametreler

element

[in] Sıralı diziye eklenecek elemanın değeri

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayShort::InsertSort(short) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ekle
   if(!array.InsertSort(100))
     {
      printf("Ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Search

Belirtilen örneğin sıralı dizideki eşitini arar.

```
int  Search(
   short  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayShort::Search(short) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.Search(100)!=-1) printf("Eleman bulundu");
   else                      printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchGreat

Sıralı dizi içinde, belirtilen örnekten daha büyük olan bir eleman arar.

```
int  SearchGreat(
   short  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayShort::SearchGreat(short) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchGreat(100)!=-1) printf("Eleman bulundu");
   else                           printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLess

Sıralı dizi içinde, belirtilen örnekten daha küçük olan bir eleman arar.

```
int  SearchLess(
   short  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayShort::SearchLess(short) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLess(100)!=-1) printf("Eleman bulundu");
   else                          printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchGreatOrEqual

Sıralı dizi içinde, belirtilen örnekten daha büyük veya eşit olan bir eleman arar.

```
int  SearchGreatOrEqual(
   short  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayShort::SearchGreatOrEqual(short) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchGreatOrEqual(100)!=-1) printf("Eleman bulundu");
   else                                  printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLessOrEqual

Sıralı dizi içinde, belirtilen örnekten daha küçük veya eşit olan bir eleman arar.

```
int  SearchLessOrEqual(
   short  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayShort::SearchLessOrEqual(short) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLessOrEqual(100)!=-1) printf("Eleman bulundu");
   else                                 printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchFirst

Sıralı dizi içinde, belirtilen kriteri sağlayan ilk elemanı arar.

```
int  SearchFirst(
   short  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayShort::SearchFirst(short) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchFirst(100)!=-1) printf("Eleman bulundu");
   else                           printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLast

Sıralı dizi içinde, belirtilen kriteri sağlayan son elemanı arar.

```
int  SearchLast(
   short  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayShort::SearchLast(short) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLast(100)!=-1) printf("Eleman bulundu");
   else                          printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLinear

Belirtilen örneğin dizi içindeki eşitini arar.

```
int  SearchLinear(
   short  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Not

Yöntem, sıralanmamış diziler için lineer arama (veya ardışık arama) algoritmasını kullanır.

Örnek:

```
//--- CArrayShort::SearchLinear(short) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- eleman ara
   if(array.SearchLinear(100)!=-1) printf("Eleman bulundu");
   else                            printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

Save

Veri dizisini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayShort::Save(int) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayShort *array=new CArrayShort;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- 100 dizi elemanı ekle
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- dosya kaydetme hatası
         printf("Dosya kaydedilemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

```
      }
   delete array;
  }
```

Load

Belirtilen dosyadan veri dizisi yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayShort::Load(int) için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayShort *array=new CArrayShort;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- dosya yükleme hatası
         printf("Dosya yüklenemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- dizi elemanlarını kullan
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %d",i,array.At(i));
```

```
      }
   delete array;
  }
```

Type

Dizinin tip tanımlayıcısını alır.

```
virtual int  Type() const
```

Dönüş Değeri

Dizinin tip tanımlayıcısı (CArrayShort için 82).

Örnek:

```
//--- CArrayShort::Type() için bir örnek
#include <Arrays\ArrayShort.mqh>
//---
void OnStart()
  {
   CArrayShort *array=new CArrayShort;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dizi tipini al
   int type=array.Type();
   //--- diziyi sil
   delete array;
  }
```

CArrayInt

CArrayInt sınıfı, int veya uint tipli değişkenler için bir dinamik dizi sınıfıdır.

Açıklama

CArrayInt sınıfı, int veya uint tipli değişkenlerden oluşan dinamik dizilerle çalışma olanağı sağlar. Sınıf içerisinde; sıralama, sıralanmış dizide arama, eleman silme/ekleme gibi uygulamalara yer verilmiştir. Ayrıca dosya işlemleri için gereken yöntemler de sınıf içerisinde yer almaktadır.

Bildirim

```
class CArrayInt : public CArray
```

Başlık

```
#include <Arrays\ArrayInt.mqh>
```

Kalıtım hiyerarşisi

CObject

CArray

CArrayInt

İlk nesil

CSpreadBuffer

Sınıf Yöntemleri

Bellek denetimi	
Reserve	Dizi boyutunun artırılması için bellek tahsis eder
Resize	Dizi için daha küçük bir boyut ayarlar
Shutdown	Belleği boşaltarak diziyi siler
Ekleme yöntemleri	
Add	Dizinin sonuna bir eleman ekler
AddArray	Başka bir diziden alınan elemanları dizinin sonuna ekler
AddArray	Başka bir diziden alınan elemanları dizinin sonuna ekler
Insert	Dizinin belirtilen konumuna eleman ekler
InsertArray	Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler
InsertArray	Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler
AssignArray	Dizinin elemanlarını başka bir diziden kopyalar
AssignArray	Dizinin elemanlarını başka bir diziden kopyalar

Bellek denetimi	
Güncelleme yöntemleri	
Update	Dizinin belirtilen konumundaki elemanı değiştirir
Shift	Dizi içindeki bir elemanı belirtilen konuma kaydırır
Silme yöntemleri	
Delete	Dizinin belirtilen konumundaki elemanı siler
DeleteRange	Dizinin belirtilen konumundaki elemanlar grubunu siler
Erişim yöntemleri	
At	Dizinin belirtilen konumundaki elemanı alır
Karşılaştırma yöntemleri	
CompareArray	Diziyi başka bir diziyle karşılaştırır
CompareArray	Diziyi başka bir diziyle karşılaştırır
Sıralı dizi işlemleri	
InsertSort	Sıralı diziye eleman ekler
Search	Sıralı dizi içinde, belirtilen örneğe eşit olan bir eleman arar
SearchGreat	Sıralı dizi içinde, belirtilen örnekten daha büyük olan bir eleman arar
SearchLess	Sıralı dizi içinde, belirtilen örnekten daha küçük olan bir eleman arar
SearchGreatOrEqual	Sıralı dizi içinde, belirtilen örnekten daha büyük veya eşit olan bir eleman arar
SearchLessOrEqual	Sıralı dizi içinde, belirtilen örnekten daha küçük veya eşit olan bir eleman arar
SearchFirst	Sıralı dizi içinde, belirtilen kriteri sağlayan ilk elemanı arar
SearchLast	Sıralı dizi içinde, belirtilen kriteri sağlayan son elemanı arar
SearchLinear	Belirtilen örneğin dizi içindeki eşitini arar
Girdi/çıktı	
virtual Save	Veri dizisini dosyaya kaydeder
virtual Load	Veri dizisini dosyadan alır
virtual Type	Dizinin tip tanımlayıcısını alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Reserve

Dizi boyutunun artırılması için bellek tahsis eder.

```
bool  Reserve(
   int  size      // eleman sayısı
   )
```

Parametreler

size

[in] Eklenecek elemanların sayısı.

Dönüş Değeri

Başarılı ise 'true', eleman sayısı sıfır veya daha küçük bir değerle ayarlanmaya çalışılmışsa 'false'.

Not

Bellek bölünmesini azaltmak için, daha önce Step (int) yöntemi ile ayarlanmış olan büyüklük değerini artırın (varsayılan değer 16).

Örnek:

```
//--- CArrayInt::Reserve(int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- bellek tahsis et
   if(!array.Reserve(1024))
     {
      printf("Tahsis hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Resize

Dizi için daha küçük bir boyut ayarlar.

```
bool  Resize(
   int  size      // eleman sayısı
   )
```

Parametreler

size

[in] Yeni dizi boyutu.

Dönüş Değeri

Başarılı ise 'true', boyut değeri sıfır veya daha küçük bir sayıyla ayarlanmaya çalışılmışsa 'false'.

Not

Dizi boyutunun değiştirilmesi belleğin etkin kullanımını sağlar. sağ taraftaki gereksiz elemanlar kaybedilir. Bellek bölünmesini azaltmak için, daha önce Step (int) yöntemi ile ayarlanmış olan büyüklük değerini azaltın (varsayılan değer 16).

Örnek:

```
//--- CArrayInt::Resize(int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi yeniden boyutlandır
   if(!array.Resize(10))
     {
      printf("Boyutlandırma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Shutdown

Belleği boşaltarak diziyi siler.

```
bool  Shutdown()
```

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayInt::Shutdown() için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi kapat
   if(!array.Shutdown())
     {
      printf("Kapatma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Add

Dizinin sonuna bir eleman ekler.

```
bool  Add(
   int  element      // eklenecek eleman
   )
```

Parametreler

element

[in] Diziye eklenecek elemanın değeri.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayInt::Add(int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Eleman ekleme hatası");
         delete array;
         return;
        }
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AddArray

Başka bir diziden alınan elemanları dizinin sonuna ekler.

```
bool  AddArray(
   const int&  src[]      // kaynak dizi
   )
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayInt::AddArray(const int &[]) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- yeni dizi ekle
   if(!array.AddArray(src))
     {
      printf("Dizi ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AddArray

Başka bir diziden alınan elemanları dizinin sonuna ekler.

```
bool  AddArray(
   const CArrayInt*  src      // kaynağın işaretçisi
   )
```

Parametreler

src

[in] Bir CArrayInt sınıf örneğinin (eklenecek kaynak elemanlar) işaretçisi.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayInt::AddArray(const CArrayInt*) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- yeni dizi ekle
   if(!array.AddArray(src))
     {
      printf("Dizi ekleme hatası");
      delete src;
      delete array;
      return;
     }
   //--- kaynak diziyi sil
   delete src;
```

```
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Insert

Dizinin belirtilen konumuna eleman ekler.

```
bool  Insert(
   int  element,     // eklenecek eleman
   int  pos          // konum
   )
```

Parametreler

element

[in] Diziye eklenecek elemanın değeri

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayInt::Insert(int,int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- eleman ekle
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Ekleme hatası");
         delete array;
         return;
        }
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

InsertArray

Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler.

```
bool  InsertArray(
   const int&  src[],     // kaynak dizi
   int          pos        // konum
   )
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', elemanlar eklenemezse 'false'.

Örnek:

```
//--- CArrayInt::InsertArray(const int &[],int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ekle
   if(!array.InsertArray(src,0))
     {
      printf("dizi ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

InsertArray

Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler.

```
bool  InsertArray(
   CArrayInt*  src,      // kaynağın işaretçisi
   int          pos      // konum
   )
```

Parametreler

src

[in] Bir CArrayInt sınıf örneğinin (eklenecek kaynak elemanlar) işaretçisi.

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', elemanlar eklenemezse 'false'.

Örnek:

```
//--- CArrayInt::InsertArray(const CArrayInt*,int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ekle
   if(!array.InsertArray(src,0))
     {
      printf("dizi ekleme hatası");
      delete src;
      delete array;
```

```
      return;
     }
   //--- kaynak diziyi sil
   delete src;
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AssignArray

Dizinin elemanlarını başka bir diziden kopyalar

```
bool  AssignArray(
   const int&  src[]      // kaynak dizi
   )
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

Dönüş Değeri

Başarılı ise 'true', elemanlar kopyalanamazsa 'false'.

Örnek:

```
//--- CArrayInt::AssignArray(const int &[]) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ata
   if(!array.AssignArray(src))
     {
      printf("Dizi atama hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AssignArray

Dizinin elemanlarını başka bir diziden kopyalar

```
bool  AssignArray(
   const CArrayInt*  src      // kaynağın işaretçisi
   )
```

Parametreler

src

[in] Bir CArrayInt sınıf örneğinin (kopyalanacak kaynak elemanlar) işaretçisi.

Dönüş Değeri

Başarılı ise 'true', elemanlar kopyalanamazsa 'false'.

Örnek:

```
//--- CArrayInt::AssignArray(const CArrayInt*) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayInt *src  =new CArrayInt;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ata
   if(!array.AssignArray(src))
     {
      printf("Dizi atama hatası");
      delete src;
      delete array;
      return;
     }
```

```
   //--- diziler aynı
   //--- kaynak diziyi sil
   delete src;
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Update

Dizinin belirtilen konumundaki elemanı değiştirir.

```
bool  Update(
   int  pos,          // konum
   int  element       //    float  element      // değer
   )
```

Parametreler

pos

[in] Değiştirilecek elemanın dizideki konumu.

element

[in] Elemanın yeni değeri

Dönüş Değeri

Başarılı ise 'true', eleman değiştirilemezse 'false'.

Örnek:

```
//--- CArrayInt::Update(int,int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı güncelle
   if(!array.Update(0,10000))
     {
      printf("Güncelleme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Shift

Dizi içindeki bir elemanı belirtilen konuma kaydırır.

```
bool  Shift(
   int  pos,        // konum
   int  shift       // Kaydırma değeri
   )
```

Parametreler

pos

[in] Kaydırılacak dizi elemanının konumu

shift

[in] Kaydırma değeri (pozitif veya negatif).

Dönüş Değeri

Başarılı ise 'true', eleman kaydırılamadıysa 'false'.

Örnek:

```
//--- CArrayInt::Shift(int,int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı kaydır
   if(!array.Shift(10,-5))
     {
      printf("Kaydırma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Delete

Dizinin belirtilen konumundaki elemanı siler.

```
bool  Delete(
   int  pos      // konum
   )
```

Parametreler

pos

[in] Silinecek elemanın konumu.

Dönüş Değeri

Başarılı ise 'true', eleman silinemezse 'false'.

Örnek:

```
//--- CArrayInt::Delete(int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı sil
   if(!array.Delete(0))
     {
      printf("Silme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

DeleteRange

Dizinin belirtilen konumundaki elemanlar grubunu siler.

```
bool  DeleteRange(
   int  from,     // ilk elemanın konumu
   int  to        // son elemanın konumu
   )
```

Parametreler

from

[in] Silinecek ilk elemanın konumu.

to

[in] Silinecek son elemanın konumu.

Dönüş Değeri

Başarılı ise 'true', elemanlar silinemezse 'false'.

Örnek:

```
//--- CArrayInt::DeleteRange(int,int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanları sil
   if(!array.DeleteRange(0,10))
     {
      printf("Silme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

At

Dizinin belirtilen konumundaki elemanı alır.

```
int  At(
   int  pos      // konum
   ) const
```

Parametreler

pos

[in] Seçilen elemanın dizi içindeki konumu.

Dönüş Değeri

Başarı durumunda elemanın değerine, olmayan bir konumdaki bir elemanın istenmesi durumunda ise INT_MAX değerine dönüş yapar (son hata değeri ERR_OUT_OF_RANGE).

Not

INT_MAX değeri geçerli bir dizi elemanı olabileceği için her zaman son hata kodunun istenmesi gerekir.

Örnek:

```
//--- CArrayInt::At(int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      int result=array.At(i);
      if(result==INT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- dizi okuma hatası
         printf("Eleman alınamadı, hata");
         delete array;
         return;
        }
      //--- elemanı kullan
      //--- . . .
     }
```

```
   //--- diziyi sil
   delete array;
  }
```

CompareArray

Diziyi başka bir diziyle karşılaştırır.

```
bool  CompareArray(
   const int&  src[]      // kaynak dizi
   ) const
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

Dönüş Değeri

Diziler eşit ise 'true', aksi durumda 'false'

Örnek:

```
//--- CArrayInt::CompareArray(const int &[]) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
int src[];
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ile karşılaştır
   int result=array.CompareArray(src);
   //--- diziyi sil
   delete array;
  }
```

CompareArray

Diziyi başka bir diziyle karşılaştırır.

```
bool  CompareArray(
   const CArrayInt*  src      // kaynağın işaretçisi
   ) const
```

Parametreler

src

[in] Bir CArrayInt sınıf örneğinin (karşılaştırılacak elemanlar) işaretçisi.

Dönüş Değeri

Diziler eşit ise 'true', aksi durumda 'false'

Örnek:

```
//--- CArrayInt::CompareArray(const CArrayInt*) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayInt *src=new CArrayInt;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ile karşılaştır
   int result=array.CompareArray(src);
   //--- dizileri sil
   delete src;
   delete array;
  }
```

InsertSort

Sıralı diziye eleman ekler.

```
bool  InsertSort(
   int  element      // eklenecek eleman
   )
```

Parametreler

element

[in] Sıralı diziye eklenecek elemanın değeri

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayInt::InsertSort(int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ekle
   if(!array.InsertSort(10000))
     {
      printf("Ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Search

Belirtilen örneğin sıralı dizideki eşitini arar.

```
int  Search(
   int  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayInt::Search(int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.Search(10000)!=-1) printf("Eleman bulundu");
   else                        printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchGreat

Sıralı dizi içinde, belirtilen örnekten daha büyük olan bir eleman arar.

```
int  SearchGreat(
   int  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayInt::SearchGreat(int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchGreat(10000)!=-1) printf("Eleman bulundu");
   else                             printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLess

Sıralı dizi içinde, belirtilen örnekten daha küçük olan bir eleman arar.

```
int  SearchLess(
   int  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayInt::SearchLess(int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLess(10000)!=-1) printf("Eleman bulundu");
   else                            printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchGreatOrEqual

Sıralı dizi içinde, belirtilen örnekten daha büyük veya eşit olan bir eleman arar.

```
int  SearchGreatOrEqual(
   int  element      // aranacak eleman
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayInt::SearchGreatOrEqual(int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchGreatOrEqual(10000)!=-1) printf("Eleman bulundu");
   else                                    printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLessOrEqual

Sıralı dizi içinde, belirtilen örnekten daha küçük veya eşit olan bir eleman arar.

```
int  SearchLessOrEqual(
   int  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayInt::SearchLessOrEqual(int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLessOrEqual(10000)!=-1) printf("Eleman bulundu");
   else                                   printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchFirst

Sıralı dizi içinde, belirtilen kriteri sağlayan ilk elemanı arar.

```
int  SearchFirst(
   int  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayInt:: SearchFirst(int) için bir örnek.
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchFirst(10000)!=-1) printf("Eleman bulundu");
   else                             printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLast

Sıralı dizi içinde, belirtilen kriteri sağlayan son elemanı arar.

```
int  SearchLast(
   int  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayInt::SearchLast(int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLast(10000)!=-1) printf("Eleman bulundu");
   else                            printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLinear

Belirtilen örneğin dizi içindeki eşitini arar.

```
int  SearchLinear(
   int  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Not

Yöntem, sıralanmamış diziler için lineer arama (veya ardışık arama) algoritmasını kullanır.

Örnek:

```
//--- CArrayInt::SearchLinear(int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- eleman ara
   if(array.SearchLinear(10000)!=-1) printf("Eleman bulundu");
   else                              printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

Save

Veri dizisini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayInt::Save(int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayInt *array=new CArrayInt;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- 100 dizi elemanı ekle
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- dosya kaydetme hatası
         printf("Dosya kaydedilemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

```
      }
   delete array;
  }
```

Load

Belirtilen dosyadan veri dizisi yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayInt::Load(int) için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayInt *array=new CArrayInt;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- dosya yükleme hatası
         printf("Dosya yüklenemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- dizi elemanlarını kullan
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %d",i,array.At(i));
```

```
      }
   delete array;
  }
```

Type

Dizinin tip tanımlayıcısını alır.

```
virtual int  Type() const
```

Dönüş Değeri

Dizinin tip tanımlayıcısı (CArrayInt için 82).

Örnek:

```
//--- CArrayInt::Type() için bir örnek
#include <Arrays\ArrayInt.mqh>
//---
void OnStart()
  {
   CArrayInt *array=new CArrayInt;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dizi tipini al
   int type=array.Type();
   //--- diziyi sil
   delete array;
  }
```

CArrayLong

CArrayLong sınıfı, long ve ulong tipli değişkenler için bir dinamik dizi sınıfıdır.

Açıklama

CArrayLong sınıfı, long ve ulong tipli değişkenlerden oluşan dinamik dizilerle çalışma olanağı sağlar. Sınıf içerisinde; sıralama, sıralanmış dizide arama, eleman silme/ekleme gibi uygulamalara yer verilmiştir. Ayrıca dosya işlemleri için gereken yöntemler de sınıf içerisinde yer almaktadır.

Bildirim

```
class CArrayLong : public CArray
```

Başlık

```
#include <Arrays\ArrayLong.mqh>
```

Kalıtım hiyerarşisi

CObject

CArray

CArrayLong

İlk nesil

CRealVolumeBuffer, CTickVolumeBuffer, CTimeBuffer

Sınıf Yöntemleri

Bellek denetimi	
Reserve	Dizi boyutunun artırılması için bellek tahsis eder
Resize	Dizi için daha küçük bir boyut ayarlar
Shutdown	Belleği boşaltarak diziyi siler
Ekleme yöntemleri	
Add	Dizinin sonuna bir eleman ekler
AddArray	Başka bir diziden alınan elemanları dizinin sonuna ekler
AddArray	Başka bir diziden alınan elemanları dizinin sonuna ekler
Insert	Dizinin belirtilen konumuna eleman ekler
InsertArray	Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler
InsertArray	Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler
AssignArray	Dizinin elemanlarını başka bir diziden kopyalar
AssignArray	Dizinin elemanlarını başka bir diziden kopyalar

Bellek denetimi	
Güncelleme yöntemleri	
Update	Dizinin belirtilen konumundaki elemanı değiştirir
Shift	Dizi içindeki bir elemanı belirtilen konuma kaydırır
Silme yöntemleri	
Delete	Dizinin belirtilen konumundaki elemanı siler
DeleteRange	Dizinin belirtilen konumundaki elemanlar grubunu siler
Erişim yöntemleri	
At	Dizinin belirtilen konumundaki elemanı alır
Karşılaştırma yöntemleri	
CompareArray	Diziyi başka bir diziyle karşılaştırır
CompareArray	Diziyi başka bir diziyle karşılaştırır
Sıralı dizi işlemleri	
InsertSort	Sıralı diziye eleman ekler
Search	Sıralı dizi içinde, belirtilen örneğe eşit olan bir eleman arar
SearchGreat	Sıralı dizi içinde, belirtilen örnekten daha büyük olan bir eleman arar
SearchLess	Sıralı dizi içinde, belirtilen örnekten daha küçük olan bir eleman arar
SearchGreatOrEqual	Sıralı dizi içinde, belirtilen örnekten daha büyük veya eşit olan bir eleman arar
SearchLessOrEqual	Sıralı dizi içinde, belirtilen örnekten daha küçük veya eşit olan bir eleman arar
SearchFirst	Sıralı dizi içinde, belirtilen kriteri sağlayan ilk elemanı arar
SearchLast	Sıralı dizi içinde, belirtilen kriteri sağlayan son elemanı arar
SearchLinear	Belirtilen örneğin dizi içindeki eşitini arar
Girdi/çıktı	
virtual Save	Veri dizisini dosyaya kaydeder
virtual Load	Veri dizisini dosyadan alır
virtual Type	Dizinin tip tanımlayıcısını alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Reserve

Dizi boyutunun artırılması için bellek tahsis eder.

```
bool  Reserve(
   int  size      // eleman sayısı
   )
```

Parametreler

size

[in] Eklenecek elemanların sayısı.

Dönüş Değeri

Başarılı ise 'true', eleman sayısı sıfır veya daha küçük bir değerle ayarlanmaya çalışılmışsa 'false'.

Not

Bellek bölünmesini azaltmak için, daha önce Step (int) yöntemi ile ayarlanmış olan büyüklük değerini artırın (varsayılan değer 16).

Örnek:

```
//--- CArrayLong::Reserve(int) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- bellek tahsis et
   if(!array.Reserve(1024))
     {
      printf("Tahsis hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Resize

Dizi için daha küçük bir boyut ayarlar.

```
bool  Resize(
   int  size      // Boyut
   )
```

Parametreler

size

[in] Yeni dizi boyutu.

Dönüş Değeri

Başarılı ise 'true', boyut değeri sıfır veya daha küçük bir sayıyla ayarlanmaya çalışılmışsa 'false'.

Not

Dizi boyutunun değiştirilmesi belleğin etkin kullanımını sağlar. sağ taraftaki gereksiz elemanlar kaybedilir. Bellek bölünmesini azaltmak için, daha önce Step (int) yöntemi ile ayarlanmış olan büyüklük değerini azaltın (varsayılan değer 16).

Örnek:

```
//--- CArrayLong::Resize(int) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi yeniden boyutlandır
   if(!array.Resize(10))
     {
      printf("Boyutlandırma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Shutdown

Belleği boşaltarak diziyi siler.

```
bool  Shutdown()
```

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayLong::Shutdown() için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi kapat
   if(!array.Shutdown())
     {
      printf("Kapatma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Add

Dizinin sonuna bir eleman ekler.

```
bool  Add(
   long  element      // eklenecek eleman
   )
```

Parametreler

element

[in] Diziye eklenecek elemanın değeri.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayLong::Add(long) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Eleman ekleme hatası");
         delete array;
         return;
        }
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AddArray

Başka bir diziden alınan elemanları dizinin sonuna ekler.

```
bool  AddArray(
   const long&  src[]      // kaynak dizi
   )
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayLong::AddArray(const long &[]) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- yeni dizi ekle
   if(!array.AddArray(src))
     {
      printf("Dizi ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AddArray

Başka bir diziden alınan elemanları dizinin sonuna ekler.

```
bool  AddArray(
   const CArrayLong*  src      // kaynağın işaretçisi
   )
```

Parametreler

src

[in] Bir CArrayLong sınıf örneğinin (eklenecek kaynak elemanlar) işaretçisi.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayLong::AddArray(const CArrayLong*) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- yeni dizi ekle
   if(!array.AddArray(src))
     {
      printf("Dizi ekleme hatası");
      delete src;
      delete array;
      return;
     }
   //--- kaynak diziyi sil
   delete src;
```

```
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Insert

Dizinin belirtilen konumuna eleman ekler.

```
bool  Insert(
   long  element,     // eklenecek eleman
   int   pos          // konum
   )
```

Parametreler

element

[in] Diziye eklenecek elemanın değeri

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayLong::Insert(long,int) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- eleman ekle
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Ekleme hatası");
         delete array;
         return;
        }
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

InsertArray

Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler.

```
bool  InsertArray(
   const long&  src[],     // kaynak dizi
   int           pos        // konum
   )
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', elemanlar eklenemezse 'false'.

Örnek:

```
//--- CArrayLong::InsertArray(const long &[],int) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ekle
   if(!array.InsertArray(src,0))
     {
      printf("dizi ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

InsertArray

Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler.

```
bool  InsertArray(
   CArrayLong*  src,      // kaynağın işaretçisi
   int           pos      //    int  pos      // konum
   )
```

Parametreler

src

[in] Bir CArrayLong sınıf örneğinin (eklenecek kaynak elemanlar) işaretçisi.

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', elemanlar eklenemezse 'false'.

Örnek:

```
//--- CArrayLong::InsertArray(const CArrayLong*,int) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ekle
   if(!array.InsertArray(src,0))
     {
      printf("dizi ekleme hatası");
      delete src;
      delete array;
```

```
      return;
     }
   //--- kaynak diziyi sil
   delete src;
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AssignArray

Dizinin elemanlarını başka bir diziden kopyalar

```
bool  AssignArray(
   const long&  src[]      // kaynak dizi
   )
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

Dönüş Değeri

Başarılı ise 'true', elemanlar kopyalanamazsa 'false'.

Örnek:

```
//--- CArrayLong::AssignArray(const long &[]) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ata
   if(!array.AssignArray(src))
     {
      printf("Dizi atama hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AssignArray

Dizinin elemanlarını başka bir diziden kopyalar

```
bool  AssignArray(
   const CArrayLong*  src      // kaynağın işaretçisi
   )
```

Parametreler

src

[in] Bir CArrayLong sınıf örneğinin (kopyalanacak kaynak elemanlar) işaretçisi.

Dönüş Değeri

Başarılı ise 'true', elemanlar kopyalanamazsa 'false'.

Örnek:

```
//--- CArrayLong::AssignArray(const CArrayLong*) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayLong *src  =new CArrayLong;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ata
   if(!array.AssignArray(src))
     {
      printf("Dizi atama hatası");
      delete src;
      delete array;
      return;
     }
   //--- diziler aynı
   //--- kaynak diziyi sil
```

```
   delete src;
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Update

Dizinin belirtilen konumundaki elemanı değiştirir.

```
bool  Update(
   int   pos,          // konum
   long  element       // değer
   )
```

Parametreler

pos

[in] Değiştirilecek elemanın dizideki konumu

element

[in] Elemanın yeni değeri

Dönüş Değeri

Başarılı ise 'true', eleman değiştirilemezse 'false'.

Örnek:

```
//--- CArrayLong::Update(int,long) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı güncelle
   if(!array.Update(0,1000000))
     {
      printf("Güncelleme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Shift

Dizi içindeki bir elemanı belirtilen konuma kaydırır.

```
bool  Shift(
   int  pos,        // konum
   int  shift       // Kaydırma değeri
   )
```

Parametreler

pos

[in] Kaydırılacak dizi elemanının konumu

shift

[in] Kaydırma değeri (pozitif veya negatif).

Dönüş Değeri

Başarılı ise 'true', eleman kaydırılamadıysa 'false'.

Örnek:

```
//--- CArrayLong::Shift(int,int) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı kaydır
   if(!array.Shift(10,-5))
     {
      printf("Kaydırma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Delete

Dizinin belirtilen konumundaki elemanı siler.

```
bool  Delete(
   int  pos      // konum
   )
```

Parametreler

pos

[in] Silinecek elemanın konumu.

Dönüş Değeri

Başarılı ise 'true', eleman silinemezse 'false'.

Örnek:

```
//--- CArrayLong::Delete(int) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı sil
   if(!array.Delete(0))
     {
      printf("Silme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

DeleteRange

Dizinin belirtilen konumundaki elemanlar grubunu siler.

```
bool  DeleteRange(
   int  from,     // ilk elemanın konumu
   int  to        // son elemanın konumu
   )
```

Parametreler

from

[in] Silinecek ilk elemanın konumu.

to

[in] Silinecek son elemanın konumu.

Dönüş Değeri

Başarılı ise 'true', elemanlar silinemezse 'false'.

Örnek:

```
//--- CArrayLong::DeleteRange(int,int) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanları sil
   if(!array.DeleteRange(0,10))
     {
      printf("Silme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

At

Dizinin belirtilen konumundaki elemanı alır.

```
long  At(
   int  pos      // konum
   ) const
```

Parametreler

pos

[in] Seçilen elemanın dizi içindeki konumu.

Dönüş Değeri

Başarı durumunda elemanın değerine, olmayan bir konumdaki bir elemanın istenmesi durumunda ise LONG_MAX değerine dönüş yapar (son hata değeri ERR_OUT_OF_RANGE).

Not

LONG_MAX değeri geçerli bir dizi elemanı olabileceği için her zaman son hata kodunun istenmesi gerekir.

Örnek:

```
//--- CArrayLong::At(int) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      long result=array.At(i);
      if(result==LONG_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- dizi okuma hatası
         printf("Eleman alınamadı, hata");
         delete array;
         return;
        }
      //--- elemanı kullan
      //--- . . .
     }
```

```
   //--- diziyi sil
   delete array;
  }
```

CompareArray

Diziyi başka bir diziyle karşılaştırır.

```
bool  CompareArray(
   const long&  src[]      // kaynak dizi
   ) const
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

Dönüş Değeri

Diziler eşit ise 'true', aksi durumda 'false'

Örnek:

```
//--- CArrayLong::CompareArray(const long &[]) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
long src[];
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ile karşılaştır
   int result=array.CompareArray(src);
   //--- diziyi sil
   delete array;
  }
```

CompareArrayconst

Diziyi başka bir diziyle karşılaştırır.

```
bool  CompareArrayconst(
   const CArrayLong*  src      // kaynağın işaretçisi
   ) const
```

Parametreler

src

[in] Bir CArrayLong sınıf örneğinin (karşılaştırılacak elemanlar) işaretçisi.

Dönüş Değeri

Diziler eşit ise 'true', aksi durumda 'false'

Örnek:

```
//--- CArrayLong::CompareArray(const CArrayLong*) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayLong *src=new CArrayLong;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ile karşılaştır
   int result=array.CompareArray(src);
   //--- dizileri sil
   delete src;
   delete array;
  }
```

InsertSort

Sıralı diziye eleman ekler.

```
bool  InsertSort(
   long  element      // eklenecek eleman
   )
```

Parametreler

element

[in] Sıralı diziye eklenecek elemanın değeri

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayLong::InsertSort(long) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ekle
   if(!array.InsertSort(1000000))
     {
      printf("Ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Search

Sıralı dizi içinde, belirtilen örneğe eşit olan bir eleman arar.

```
int  Search(
   long  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayLong::Search(long) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.Search(1000000)!=-1) printf("Eleman bulundu");
   else                          printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchGreat

Sıralı dizi içinde, belirtilen örnekten daha büyük olan bir eleman arar.

```
int  SearchGreat(
   long  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayLong::SearchGreat(long) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchGreat(1000000)!=-1) printf("Eleman bulundu");
   else                               printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLess

Sıralı dizi içinde, belirtilen örnekten daha küçük olan bir eleman arar.

```
int  SearchLess(
   long  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayLong::SearchLess(long) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLess(1000000)!=-1) printf("Eleman bulundu");
   else                              printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchGreatOrEqual

Sıralı dizi içinde, belirtilen örnekten daha büyük veya eşit olan bir eleman arar.

```
int  SearchGreatOrEqual(
   long  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayLong::SearchGreatOrEqual(long) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchGreatOrEqual(1000000)!=-1) printf("Eleman bulundu");
   else                                      printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLessOrEqual

Sıralı dizi içinde, belirtilen örnekten daha küçük veya eşit olan bir eleman arar.

```
int  SearchLessOrEqual(
   long  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayLong::SearchLessOrEqual(long) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLessOrEqual(1000000)!=-1) printf("Eleman bulundu");
   else                                     printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchFirst

Sıralı dizi içinde, belirtilen kriteri sağlayan ilk elemanı arar.

```
int  SearchFirst(
   long  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayLong::SearchFirst(long) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchFirst(1000000)!=-1) printf("Eleman bulundu");
   else                               printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLast

Sıralı dizi içinde, belirtilen kriteri sağlayan son elemanı arar.

```
int  SearchLast(
   long  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayLong::SearchLast(long) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLast(1000000)!=-1) printf("Eleman bulundu");
   else                              printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLinear

Dizi içinde, belirtilen kriteri sağlayan bir eleman arar.

```
int  SearchLinear(
   long  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Not

Yöntem, sıralanmamış diziler için lineer arama (veya ardışık arama) algoritmasını kullanır.

Örnek:

```
//--- CArrayLong::SearchLinear(long) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- eleman ara
   if(array.SearchLinear(1000000)!=-1) printf("Eleman bulundu");
   else                                printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

Save

Veri dizisini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayLong::Save(int) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayLong *array=new CArrayLong;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- 100 dizi elemanı ekle
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- dosya kaydetme hatası
         printf("Dosya kaydedilemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

```
      }
   delete array;
  }
```

Load

Belirtilen dosyadan veri dizisi yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayLong::Load(int) için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   int          file_handle;
   CArrayLong *array=new CArrayLong;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- dosya yükleme hatası
         printf("Dosya yüklenemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- dizi elemanlarını kullan
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %I64",i,array.At(i));
```

```
      }
   delete array;
  }
```

Type

Dizinin tip tanımlayıcısını alır.

```
virtual int  Type() const
```

Dönüş Değeri

Dizinin tip tanımlayıcısı (CArrayLong için 84).

Örnek:

```
//--- CArrayLong::Type() için bir örnek
#include <Arrays\ArrayLong.mqh>
//---
void OnStart()
  {
   CArrayLong *array=new CArrayLong;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dizi tipini al
   int type=array.Type();
   //--- diziyi sil
   delete array;
  }
```

CArrayFloat

CArrayFloat sınıfı, float tipli değişkenler için bir dinamik dizi sınıfıdır.

Açıklama

CArrayFloat sınıfı, float tipli değişkenlerden oluşan dinamik dizilerle çalışma olanağı sağlar. Sınıf içerisinde; sıralama, sıralanmış dizide arama, eleman silme/ekleme gibi uygulamalara yer verilmiştir. Ayrıca dosya işlemleri için gereken yöntemler de sınıf içerisinde yer almaktadır.

Bildirim

```
class CArrayFloat : public CArray
```

Başlık

```
#include <Arrays\ArrayFloat.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayFloat

Sınıf Yöntemleri

Özellikler	
Delta	Karşılaştırma toleransını ayarlar
Bellek denetimi	
Reserve	Dizi boyutunun artırılması için bellek tahsis eder
Resize	Dizi için daha küçük bir boyut ayarlar
Shutdown	Belleği boşaltarak diziyi siler
Ekleme yöntemleri	
Add	Dizinin sonuna bir eleman ekler
AddArray	Başka bir diziden alınan elemanları dizinin sonuna ekler
AddArray	Başka bir diziden alınan elemanları dizinin sonuna ekler
Insert	Dizinin belirtilen konumuna eleman ekler
InsertArray	Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler
InsertArray	Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler
AssignArray	Dizinin elemanlarını başka bir diziden kopyalar
AssignArray	Dizinin elemanlarını başka bir diziden kopyalar

Özellikler	
Güncelleme yöntemleri	
Update	Dizinin belirtilen konumundaki elemanı değiştirir
Shift	Dizi içindeki bir elemanı belirtilen konuma kaydırır
Delete	Dizinin belirtilen konumundaki elemanı siler
DeleteRange	Dizinin belirtilen konumundaki elemanlar grubunu siler
Erişim yöntemleri	
At	Dizinin belirtilen konumundaki elemanı alır
Karşılaştırma yöntemleri	
CompareArray	Diziyi başka bir diziyle karşılaştırır
CompareArray	Diziyi başka bir diziyle karşılaştırır
Sıralı dizi işlemleri	
InsertSort	Sıralı diziye eleman ekler
Search	Sıralı dizi içinde, belirtilen örneğe eşit olan bir eleman arar
SearchGreat	Sıralı dizi içinde, belirtilen örnekten daha büyük olan bir eleman arar
SearchLess	Sıralı dizi içinde, belirtilen örnekten daha küçük olan bir eleman arar
SearchGreatOrEqual	Sıralı dizi içinde, belirtilen örnekten daha büyük veya eşit olan bir eleman arar
SearchLessOrEqual	Sıralı dizi içinde, belirtilen örnekten daha küçük veya eşit olan bir eleman arar
SearchFirst	Sıralı dizi içinde, belirtilen kriteri sağlayan ilk elemanı arar
SearchLast	Sıralı dizi içinde, belirtilen kriteri sağlayan son elemanı arar
SearchLinear	Belirtilen örneğin dizi içindeki eşitini arar
Girdi/çıktı	
virtual Save	Veri dizisini dosyaya kaydeder
virtual Load	Veri dizisini dosyadan alır
virtual Type	Dizinin tip tanımlayıcısını alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Delta

Karşılaştırma toleransını ayarlar.

```
void  Delta(
   float  delta      // Tolerans
   )
```

Parametreler

delta

[in] Yeni karşılaştırma toleransı değeri.

Dönüş Değeri

None

Not

Arama için kabul edilebilecek tolerans değerini ayarlar. Değerlerin mutlak farkı bu toleranstan büyük olmadıkça değerler eşit kabul edilir. Varsayılan tolerans değeri 0.0 olarak ayarlanmıştır.

Örnek:

```
//--- CArrayFloat::Delta(float) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- karşılaştırma toleransını ayarla
   array.Delta(0.001);
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Reserve

Dizi boyutunun artırılması için bellek tahsis eder.

```
bool  Reserve(
   int  size      // eleman sayısı
   )
```

Parametreler

size

[in] Eklenecek elemanların sayısı.

Dönüş Değeri

Başarılı ise 'true', eleman sayısı sıfır veya daha küçük bir değerle ayarlanmaya çalışılmışsa 'false'.

Not

Bellek bölünmesini azaltmak için, daha önce Step (int) yöntemi ile ayarlanmış olan büyüklük değerini artırın (varsayılan değer 16).

Örnek:

```
//--- CArrayFloat::Reserve(int) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- bellek tahsis et
   if(!array.Reserve(1024))
     {
      printf("Tahsis hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Resize

Dizi için daha küçük bir boyut ayarlar.

```
bool  Resize(
   int  size      // Boyut
   )
```

Parametreler

size

[in] Yeni dizi boyutu.

Dönüş Değeri

Başarılı ise 'true', boyut değeri sıfır veya daha küçük bir sayıyla ayarlanmaya çalışılmışsa 'false'.

Not

Dizi boyutunun değiştirilmesi belleğin etkin kullanımını sağlar. sağ taraftaki gereksiz elemanlar kaybedilir. Bellek bölünmesini azaltmak için, daha önce Step (int) yöntemi ile ayarlanmış olan büyüklük değerini azaltın (varsayılan değer 16).

Örnek:

```
//--- CArrayFloat::Resize(int) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi yeniden boyutlandır
   if(!array.Resize(10))
     {
      printf("Boyutlandırma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Shutdown

Belleği boşaltarak diziyi siler.

```
bool  Shutdown()
```

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayFloat::Shutdown() için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi kapat
   if(!array.Shutdown())
     {
      printf("Kapatma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Add

Dizinin sonuna bir eleman ekler.

```
bool  Add(
   float  element      // eklenecek eleman
   )
```

Parametreler

element

[in] Diziye eklenecek elemanın değeri.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayFloat::Add(float) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Eleman ekleme hatası");
         delete array;
         return;
        }
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AddArray

Başka bir diziden alınan elemanları dizinin sonuna ekler.

```
bool  AddArray(
   const float&  src[]      // kaynak dizi
   )
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayFloat::AddArray(const float &[]) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- yeni dizi ekle
   if(!array.AddArray(src))
     {
      printf("Dizi ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AddArray

Başka bir diziden alınan elemanları dizinin sonuna ekler.

```
bool  AddArray(
   const CArrayFloat*  src      // kaynağın işaretçisi
   )
```

Parametreler

src

[in] Bir CArrayFloat sınıf örneğinin (eklenecek kaynak elemanlar) işaretçisi.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayFloat::AddArray(const CArrayFloat*) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- yeni dizi ekle
   if(!array.AddArray(src))
     {
      printf("Dizi ekleme hatası");
      delete src;
      delete array;
      return;
     }
   //--- kaynak diziyi sil
   delete src;
```

```
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Insert

Dizinin belirtilen konumuna eleman ekler.

```
bool  Insert(
   float  element,      // eklenecek eleman
   int    pos           // konum
   )
```

Parametreler

element

[in] Diziye eklenecek elemanın değeri

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayFloat::Insert(float,int) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- eleman ekle
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Ekleme hatası");
         delete array;
         return;
        }
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

InsertArray

Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler.

```
bool  InsertArray(
   const float&  src[],     // kaynak dizisi
   int            pos        // konum
   )
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', elemanlar eklenemezse 'false'.

Örnek:

```
//--- CArrayFloat::InsertArray(const float &[],int) örneği
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ekle
   if(!array.InsertArray(src,0))
     {
      printf("dizi ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

InsertArray

Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler.

```
bool  InsertArray(
   CArrayFloat*  src,      // kaynağın işaretçisi
   int            pos      // konum
   )
```

Parametreler

src

[in] Bir CArrayFloat sınıf örneğinin (eklenecek kaynak elemanlar) işaretçisi.

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', elemanlar eklenemezse 'false'.

Örnek:

```
//--- CArrayFloat::InsertArray(const CArrayFloat*,int) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ekle
   if(!array.InsertArray(src,0))
     {
      printf("dizi ekleme hatası");
      delete src;
      delete array;
```

```
      return;
     }
   //--- kaynak diziyi sil
   delete src;
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AssignArray

Dizinin elemanlarını başka bir diziden kopyalar

```
bool  AssignArray(
   const float&  src[]      // kaynak dizi
   )
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

Dönüş Değeri

Başarılı ise 'true', elemanlar kopyalanamazsa 'false'.

Örnek:

```
//--- CArrayFloat::AssignArray(const float &[]) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ata
   if(!array.AssignArray(src))
     {
      printf("Dizi atama hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AssignArray

Dizinin elemanlarını başka bir diziden kopyalar

```
bool  AssignArray(
   const CArrayFloat*  src      // kaynağın işaretçisi
   )
```

Parametreler

src

[in] Bir CArrayFloat sınıf örneğinin (kopyalanacak kaynak elemanlar) işaretçisi.

Dönüş Değeri

Başarılı ise 'true', elemanlar kopyalanamazsa 'false'.

Örnek:

```
//--- CArrayFloat::AssignArray(const CArrayFloat*) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayFloat *src  =new CArrayFloat;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ata
   if(!array.AssignArray(src))
     {
      printf("Dizi atama hatası");
      delete src;
      delete array;
      return;
     }
   //--- diziler aynı
   //--- kaynak diziyi sil
```

```
   delete src;
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Update

Dizinin belirtilen konumundaki elemanı değiştirir.

```
bool  Update(
   int    pos,          // konum
   float  element       // değer
   )
```

Parametreler

pos

[in] Değiştirilecek elemanın dizideki konumu

element

[in] Elemanın yeni değeri

Dönüş Değeri

Başarılı ise 'true', eleman değiştirilemezse 'false'.

Örnek:

```
//--- CArrayFloat::Update(int,float) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı güncelle
   if(!array.Update(0,100.0))
     {
      printf("Güncelleme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Shift

Dizi içindeki bir elemanı belirtilen konuma kaydırır.

```
bool  Shift(
   int  pos,        // konum
   int  shift       // Kaydırma değeri
   )
```

Parametreler

pos

[in] Kaydırılacak dizi elemanının konumu

shift

[in] Kaydırma değeri (pozitif veya negatif).

Dönüş Değeri

Başarılı ise 'true', eleman kaydırılamadıysa 'false'.

Örnek:

```
//--- CArrayFloat::Shift(int,int) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı kaydır
   if(!array.Shift(10,-5))
     {
      printf("Kaydırma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Delete

Dizinin belirtilen konumundaki elemanı siler.

```
bool  Delete(
   int  pos      // konum
   )
```

Parametreler

pos

[in] Silinecek elemanın konumu.

Dönüş Değeri

Başarılı ise 'true', eleman silinemezse 'false'.

Örnek:

```
//--- CArrayFloat::Delete(int) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı sil
   if(!array.Delete(0))
     {
      printf("Silme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

DeleteRange

Dizinin belirtilen konumundaki elemanlar grubunu siler.

```
bool  DeleteRange(
   int  from,     // ilk elemanın konumu
   int  to        // son elemanın konumu
   )
```

Parametreler

from

[in] Silinecek ilk elemanın konumu.

to

[in] Silinecek son elemanın konumu.

Dönüş Değeri

Başarılı ise 'true', elemanlar silinemezse 'false'.

Örnek:

```
//--- CArrayFloat::DeleteRange(int,int) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanları sil
   if(!array.DeleteRange(0,10))
     {
      printf("Silme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

At

Dizinin belirtilen konumundaki elemanı alır.

```
float  At(
   int  pos      // konum
   ) const
```

Parametreler

pos

[in] Seçilen elemanın dizi içindeki konumu.

Dönüş Değeri

Başarı durumunda elemanın değerine, olmayan bir konumdaki bir elemanın istenmesi durumunda ise FLT_MAX değerine dönüş yapar (son hata değeri ERR_OUT_OF_RANGE).

Not

FLT_MAX değeri geçerli bir dizi elemanı olabileceği için her zaman son hata kodunun istenmesi gerekir.

Örnek:

```
//--- CArrayFloat::At(int) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      float result=array.At(i);
      if(result==FLT_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- error reading from array
         printf("Eleman alınamadı, hata");
         delete array;
         return;
        }
      //--- elemanı kullan
      //--- . . .
     }
```

```
   //--- diziyi sil
   delete array;
  }
```

CompareArray

Diziyi başka bir diziyle karşılaştırır.

```
bool  CompareArray(
   const float&  src[]      // kaynak dizi
   ) const
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

Dönüş Değeri

Diziler eşit ise 'true', aksi durumda 'false'

Örnek:

```
//--- CArrayFloat::CompareArray(const float &[]) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
float src[];
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ile karşılaştır
   int result=array.CompareArray(src);
   //--- diziyi sil
   delete array;
  }
```

AssignArrayconst

Dizinin elemanlarını başka bir diziden kopyalar

```
bool  AssignArrayconst(
   const CArrayFloat*  src      // kaynağın işaretçisi
   ) const
```

Parametreler

src

[in] Bir CArrayFloat sınıf örneğinin (kopyalanacak kaynak elemanlar) işaretçisi.

Dönüş Değeri

Başarılı ise 'true', elemanlar kopyalanamazsa 'false'.

Örnek:

```
//--- CArrayFloat::CompareArray(const CArrayFloat*) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayFloat *src=new CArrayFloat;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ile karşılaştır
   int result=array.CompareArray(src);
   //--- dizileri sil
   delete src;
   delete array;
  }
```

InsertSort

Sıralı diziye eleman ekler.

```
bool  InsertSort(
   float  element      // eklenecek eleman
   )
```

Parametreler

element

[in] Sıralı diziye eklenecek elemanın değeri

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayFloat::InsertSort(float) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ekle
   if(!array.InsertSort(100.0))
     {
      printf("Ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Search

Belirtilen örneğin sıralı dizideki eşitini arar.

```
int  Search(
   float  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayFloat::Search(float) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.Search(100.0)!=-1) printf("Eleman bulundu");
   else                        printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchGreat

Sıralı dizi içinde, belirtilen örnekten daha büyük olan bir eleman arar.

```
int  SearchGreat(
   float  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayFloat::SearchGreat(float) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchGreat(100.0)!=-1) printf("Eleman bulundu");
   else                             printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLess

Sıralı dizi içinde, belirtilen örnekten daha küçük olan bir eleman arar.

```
int  SearchLess(
   float  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayFloat:: SearchLess(float) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLess(100.0)!=-1) printf("Eleman bulundu");
   else                            printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchGreatOrEqual

Sıralı dizi içinde, belirtilen örnekten daha büyük veya eşit olan bir eleman arar.

```
int  SearchGreatOrEqual(
   float  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayFloat::SearchGreatOrEqual(float) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchGreatOrEqual(100.0)!=-1) printf("Eleman bulundu");
   else                                    printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLessOrEqual

Sıralı dizi içinde, belirtilen örnekten daha küçük veya eşit olan bir eleman arar.

```
int  SearchLessOrEqual(
   float  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayFloat::SearchLessOrEqual(float) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLessOrEqual(100.0)!=-1) printf("Eleman bulundu");
   else                                   printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchFirst

Sıralı dizi içinde, belirtilen kriteri sağlayan ilk elemanı arar.

```
int  SearchFirst(
   float  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayFloat::SearchFirst(float) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchFirst(100.0)!=-1) printf("Eleman bulundu");
   else                             printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLast

Sıralı dizi içinde, belirtilen kriteri sağlayan son elemanı arar.

```
int  SearchLast(
   float  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayFloat::SearchLast(float) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLast(100.0)!=-1) printf("Eleman bulundu");
   else                            printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLinear

Belirtilen örneğin dizi içindeki eşitini arar.

```
int  SearchLinear(
   float  element      // örnek
   ) const
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Not

Yöntem, sıralanmamış diziler için lineer arama (veya ardışık arama) algoritmasını kullanır.

Örnek:

```
//--- CArrayFloat::SearchLinear(float) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- eleman ara
   if(array.SearchLinear(100.0)!=-1) printf("Eleman bulundu");
   else                              printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

Save

Veri dizisini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayFloat::Save(int) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- 100 dizi elemanı ekle
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- dosya kaydetme hatası
         printf("Dosya kaydedilemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

```
      }
   delete array;
  }
```

Load

Belirtilen dosyadan veri dizisi yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayFloat::Load(int) için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- dosya yükleme hatası
         printf("Dosya yüklenemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- dizi elemanlarını kullan
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %f",i,array.At(i));
```

```
      }
   delete array;
  }
```

Type

Dizinin tip tanımlayıcısını alır.

```
virtual int  Type() const
```

Dönüş Değeri

Dizinin tip tanımlayıcısı (CArrayFloat için 87).

Örnek:

```
//--- CArrayFloat::Type() için bir örnek
#include <Arrays\ArrayFloat.mqh>
//---
void OnStart()
  {
   CArrayFloat *array=new CArrayFloat;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dizi tipini al
   int type=array.Type();
   //--- diziyi sil
   delete array;
  }
```

CArrayDouble

CArrayDouble sınıfı, double tipli değişkenler için bir dinamik dizi sınıfıdır.

Açıklama

CArrayDouble sınıfı, double tipli değişkenlerden oluşan dinamik dizilerle çalışma olanağı sağlar. Sınıf içerisinde; sıralama, sıralanmış dizide arama, eleman silme/ekleme gibi uygulamalara yer verilmiştir. Ayrıca dosya işlemleri için gereken yöntemler de sınıf içerisinde yer almaktadır.

Bildirim

```
class CArrayDouble : public CArray
```

Başlık

```
#include <Arrays\ArrayDouble.mqh>
```

Kalıtım hiyerarşisi

CObject

CArray

CArrayDouble

İlk nesil

CDoubleBuffer

Sınıf Yöntemleri

Özellikler	
Delta	Karşılaştırma toleransını ayarlar
Bellek denetimi	
Reserve	Dizi boyutunun artırılması için bellek tahsis eder
Resize	Dizi için daha küçük bir boyut ayarlar
Shutdown	Belleği boşaltarak diziyi siler
Ekleme yöntemleri	
Add	Dizinin sonuna bir eleman ekler
AddArray	Başka bir diziden alınan elemanları dizinin sonuna ekler
AddArray	Başka bir diziden alınan elemanları dizinin sonuna ekler
Insert	Dizinin belirtilen konumuna eleman ekler
InsertArray	Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler
InsertArray	Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler

Özellikler	
AssignArray	Dizinin elemanlarını başka bir diziden kopyalar
AssignArray	Dizinin elemanlarını başka bir diziden kopyalar
Güncelleme yöntemleri	
Update	Dizinin belirtilen konumundaki elemanı değiştirir
Shift	Dizi içindeki bir elemanı belirtilen konuma kaydırır
Silme yöntemleri	
Delete	Dizinin belirtilen konumundaki elemanı siler
DeleteRange	Dizinin belirtilen konumundaki elemanlar grubunu siler
Erişim yöntemleri	
At	Dizinin belirtilen konumundaki elemanı alır
Karşılaştırma yöntemleri	
CompareArray	Diziyi başka bir diziyle karşılaştırır
CompareArray	Diziyi başka bir diziyle karşılaştırır
Minimum/maksimum değerler için arama	
Minimum	Belirtilen dizi parçasındaki en küçük elemanın indisini alır
Maximum	Belirtilen dizi parçasındaki en büyük elemanın indisini alır
Sıralı dizi işlemleri	
InsertSort	Sıralı diziye eleman ekler
Search	Sıralı dizi içinde, belirtilen örneğe eşit olan bir eleman arar
SearchGreat	Sıralı dizi içinde, belirtilen örnekten daha büyük olan bir eleman arar
SearchLess	Sıralı dizi içinde, belirtilen örnekten daha küçük olan bir eleman arar
SearchGreatOrEqual	Sıralı dizi içinde, belirtilen örnekten daha büyük veya eşit olan bir eleman arar
SearchLessOrEqual	Sıralı dizi içinde, belirtilen örnekten daha küçük veya eşit olan bir eleman arar
SearchFirst	Sıralı dizi içinde, belirtilen kriteri sağlayan ilk elemanı arar
SearchLast	Sıralı dizi içinde, belirtilen kriteri sağlayan son elemanı arar
SearchLinear	Belirtilen örneğin dizi içindeki eşitini arar
Girdi/çıktı	
virtual Save	Veri dizisini dosyaya kaydeder

Özellikler	
virtual Load	Veri dizisini dosyadan alır
virtual Type	Dizinin tip tanımlayıcısını alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Delta

Karşılaştırma toleransını ayarlar.

```
void  Delta(
   double  delta      // Tolerans
   )
```

Parametreler

delta

[in] Yeni karşılaştırma toleransı değeri.

Dönüş Değeri

No

Not

Arama için kabul edilebilecek tolerans değerini ayarlar. Değerlerin mutlak farkı bu toleranstan büyük olmadıkça değerler eşit kabul edilir. Varsayılan tolerans değeri 0.0 olarak ayarlanmıştır.

Örnek:

```
//--- CArrayDouble::Delta(double) örneği
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- karşılaştırma toleransını ayarla
   array.Delta(0.001);
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Reserve

Dizi boyutunun artırılması için bellek tahsis eder.

```
bool  Reserve(
   int  size      // eleman sayısı
   )
```

Parametreler

size

[in] Eklenecek elemanların sayısı.

Dönüş Değeri

Başarılı ise 'true', eleman sayısı sıfır veya daha küçük bir değerle ayarlanmaya çalışılmışsa 'false'.

Not

Bellek bölünmesini azaltmak için, daha önce Step (int) yöntemi ile ayarlanmış olan büyüklük değerini artırın (varsayılan değer 16).

Örnek:

```
//--- CArrayDouble::Reserve(int) örneği
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- bellek tahsis et
   if(!array.Reserve(1024))
     {
      printf("Tahsis hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Resize

Dizi için daha küçük bir boyut ayarlar.

```
bool  Resize(
   int  size      // Boyut
   )
```

Parametreler

size

[in] Yeni dizi boyutu.

Dönüş Değeri

Başarılı ise 'true', boyut değeri sıfır veya daha küçük bir sayıyla ayarlanmaya çalışılmışsa 'false'.

Not

Dizi boyutunun değiştirilmesi belleğin etkin kullanımını sağlar. sağ taraftaki gereksiz elemanlar kaybedilir. Bellek bölünmesini azaltmak için, daha önce Step (int) yöntemi ile ayarlanmış olan büyüklük değerini azaltın (varsayılan değer 16).

Örnek:

```
//--- CArrayDouble::Resize(int) örneği
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi yeniden boyutlandır
   if(!array.Resize(10))
     {
      printf("Boyutlandırma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Shutdown

Belleği boşaltarak diziyi siler.

```
bool  Shutdown()
```

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayDouble::Shutdown() için bir örnek
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi kapat
   if(!array.Shutdown())
     {
      printf("Kapatma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Add

Dizinin sonuna bir eleman ekler.

```
bool  Add(
   double  element      // eklenecek eleman
   )
```

Parametreler

element

[in] Diziye eklenecek elemanın değeri.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayDouble::Add(double) örneği
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   for(int i=0;i<100;i++)
     {
      if(!array.Add(i))
        {
         printf("Eleman ekleme hatası");
         delete array;
         return;
        }
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AddArray

Başka bir diziden alınan elemanları dizinin sonuna ekler.

```
bool  AddArray(
   const double&  src[]      // kaynak dizi
   )
```

Parametreler

src[]

[in] Elemanları eklenecek olan kaynak dizinin referansı.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayDouble::AddArray(const double &[]) örneği
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- yeni dizi ekle
   if(!array.AddArray(src))
     {
      printf("Dizi ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AddArray

Başka bir diziden alınan elemanları dizinin sonuna ekler.

```
bool  AddArray(
   const CArrayDouble*  src      // kaynağın işaretçisi
   )
```

Parametreler

src

[in] Bir CArrayDouble sınıf örneğinin (eklenecek kaynak elemanlar) işaretçisi.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayDouble::AddArray(const CArrayDouble*) örneği
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- yeni dizi ekle
   if(!array.AddArray(src))
     {
      printf("Dizi ekleme hatası");
      delete src;
      delete array;
      return;
     }
   //--- kaynak diziyi sil
   delete src;
```

```
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Insert

Dizinin belirtilen konumuna eleman ekler.

```
bool  Insert(
   double  element,     // eklenecek eleman
   int     pos          // konum
   )
```

Parametreler

element

[in] Diziye eklenecek elemanın değeri

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayDouble::Insert(double,int) örneği
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- eleman ekle
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(i,0))
        {
         printf("Ekleme hatası");
         delete array;
         return;
        }
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

InsertArray

Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler.

```
bool  InsertArray(
   const double&  src[],      // kaynak dizi
   int            pos         // konum
   )
```

Parametreler

src[]

[in] Elemanları eklenecek olan kaynak dizinin referansı

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', elemanlar eklenemezse 'false'.

Örnek:

```
//--- CArrayDouble::InsertArray(const double &[],int) örneği
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ekle
   if(!array.InsertArray(src,0))
     {
      printf("dizi ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

InsertArray

Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler.

```
bool  InsertArray(
   CArrayDouble*  src,      // kaynağın işaretçisi
   int            pos       // konum
   )
```

Parametreler

src

[in] Bir CArrayDouble sınıf örneğinin (eklenecek kaynak elemanlar) işaretçisi.

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', elemanlar eklenemezse 'false'.

Örnek:

```
//--- CArrayDouble::InsertArray(const CArrayDouble*,int) örneği
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ekle
   if(!array.InsertArray(src,0))
     {
      printf("dizi ekleme hatası");
      delete src;
      delete array;
```

```
      return;
     }
   //--- kaynak diziyi sil
   delete src;
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AssignArray

Dizinin elemanlarını başka bir diziden kopyalar

```
bool  AssignArray(
   const double&  src[]      // kaynak dizi
   )
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

Dönüş Değeri

Başarılı ise 'true', elemanlar kopyalanamazsa 'false'.

Örnek:

```
//--- CArrayDouble::AssignArray(const double &[]) örneği
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ata
   if(!array.AssignArray(src))
     {
      printf("Dizi atama hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AssignArray

Dizinin elemanlarını başka bir diziden kopyalar

```
bool  AssignArray(
   const CArrayDouble*  src      // kaynağın işaretçisi
   )
```

Parametreler

src

[in] Bir CArrayDouble sınıf örneğinin (kopyalanacak kaynak elemanlar) işaretçisi.

Dönüş Değeri

Başarılı ise 'true', elemanlar kopyalanamazsa 'false'.

Örnek:

```
//--- CArrayDouble::AssignArray(const CArrayDouble*) örneği
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayDouble *src  =new CArrayDouble;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ata
   if(!array.AssignArray(src))
     {
      printf("Dizi atama hatası");
      delete src;
      delete array;
      return;
     }
   //--- diziler aynı
   //--- kaynak diziyi sil
```

```
   delete src;
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Update

Dizinin belirtilen konumundaki elemanı değiştirir.

```
bool  Update(
   int     pos,          // konum
   double  element       // değer
   )
```

Parametreler

pos

[in] Değiştirilecek elemanın dizideki konumu

element

[in] yeni değer.

Dönüş Değeri

Başarılı ise 'true', eleman değiştirilemezse 'false'.

Örnek:

```
//--- CArrayDouble::Update(int,double) için bir örnek
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı güncelle
   if(!array.Update(0,100.0))
     {
      printf("Güncelleme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Shift

Dizi içindeki bir elemanı belirtilen konuma kaydırır.

```
bool  Shift(
   int  pos,        // konum
   int  shift       // Kaydırma değeri
   )
```

Parametreler

pos

[in] Kaydırılacak dizi elemanının konumu

shift

[in] Kaydırma değeri (pozitif veya negatif).

Dönüş Değeri

Başarılı ise 'true', eleman kaydırılamadıysa 'false'.

Örnek:

```
//--- CArrayDouble::Shift(int,int) için bir örnek
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı kaydır
   if(!array.Shift(10,-5))
     {
      printf("Kaydırma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Delete

Dizinin belirtilen konumundaki elemanı siler.

```
bool  Delete(
   int  pos      // konum
   )
```

Parametreler

pos

[in] Silinecek elemanın konumu.

Dönüş Değeri

Başarılı ise 'true', eleman silinemezse 'false'.

Örnek:

```
//--- CArrayDouble::Delete(int) örneği
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı sil
   if(!array.Delete(0))
     {
      printf("Silme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

DeleteRange

Dizinin belirtilen konumundaki elemanlar grubunu siler.

```
bool  DeleteRange(
   int  from,     // ilk elemanın konumu
   int  to        // son elemanın konumu
   )
```

Parametreler

from

[in] Silinecek ilk elemanın konumu.

to

[in] Silinecek son elemanın konumu.

Dönüş Değeri

Başarılı ise 'true', elemanlar silinemezse 'false'.

Örnek:

```
//--- CArrayDouble::DeleteRange(int,int) örneği
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanları sil
   if(!array.DeleteRange(0,10))
     {
      printf("Silme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

At

Dizinin belirtilen konumundaki elemanı alır.

```
double  At(
   int  pos      // konum
   ) const
```

Parametreler

pos

[in] Seçilen elemanın dizi içindeki konumu.

Dönüş Değeri

Başarı durumunda elemanın değerine, olmayan bir konumdaki bir elemanın istenmesi durumunda ise DBL_MAX değerine dönüş yapar (son hata değeri ERR_OUT_OF_RANGE).

Not

DBL_MAX değeri geçerli bir dizi elemanı olabileceği için her zaman son hata kodunun istenmesi gerekir.

Örnek:

```
//--- CArrayDouble::At(int) örneği
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      double result=array.At(i);
      if(result==DBL_MAX && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //---  dizi okuma hatası
         printf("Eleman alınamadı, hata");
         delete array;
         return;
        }
      //--- elemanı kullan
      //--- . . .
     }
```

```
   //--- diziyi sil
   delete array;
  }
```

CompareArray

Diziyi başka bir diziyle karşılaştırır.

```
bool  CompareArray(
   const double&  src[]      // kaynak dizi
   ) const
```

Parametreler

src[]

[in] Karşılaştırma için kullanılacak dizinin referansı.

Dönüş Değeri

Diziler eşit ise 'true', aksi durumda 'false'

Örnek:

```
//--- CArrayDouble::CompareArray(const double &[]) örneği
#include <Arrays\ArrayDouble.mqh>
//---
double src[];
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ile karşılaştır
   int result=array.CompareArray(src);
   //--- diziyi sil
   delete array;
  }
```

CompareArray

Diziyi başka bir diziyle karşılaştırır.

```
bool  CompareArray(
   const CArrayDouble*  src      // kaynağın işaretçisi
   ) const
```

Parametreler

src

[in] Bir CArrayDouble sınıf örneğinin işaretçisi (karşılaştırma için kaynak elemanlar).

Dönüş Değeri

Diziler eşit ise 'true', aksi durumda 'false'

Örnek:

```
//--- CArrayDouble::CompareArray(const CArrayDouble*) örneği
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayDouble *src=new CArrayDouble;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ile karşılaştır
   int result=array.CompareArray(src);
   //--- dizileri sil
   delete src;
   delete array;
  }
```

Minimum

Belirtilen dizi parçasındaki en küçük elemanın indisini alır.

```
int  Minimum(
   int  start,      // başlangıç indisi
   int  count       // işlenecek elemanların sayısı
   ) const
```

Parametreler

start

[in] Dizideki başlangıç konumunun indisi.

count

[in] İşlenecek elemanların sayısı.

Dönüş değeri

Belirtilen dizi parçasındaki en küçük elemanın indisi.

Maximum

Belirtilen dizi parçasındaki en büyük elemanın indisini alır.

```
int  Maximum(
   int  start,      // başlangıç indisi
   int  count       // işlenecek elemanların sayısı
   ) const
```

Parametreler

start

[in] Dizideki başlangıç konumunun indisi.

count

[in] İşlenecek elemanların sayısı.

Dönüş değeri

Belirtilen dizi parçasındaki en büyük elemanın indisi.

InsertSort

Sıralı diziye eleman ekler.

```
bool  InsertSort(
   double  element      // eklenecek eleman
   )
```

Parametreler

element

[in] Sıralı diziye eklenecek elemanın değeri

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayDouble::InsertSort(double) örneği
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ekle
   if(!array.InsertSort(100.0))
     {
      printf("Ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Search

Belirtilen örneğin sıralı dizideki eşitini arar.

```
int  Search(
   double  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayDouble::Search(double) için bir örnek
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.Search(100.0)!=-1) printf("Eleman bulundu");
   else                        printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchGreat

Sıralı dizi içinde, belirtilen örnekten daha büyük olan bir eleman arar.

```
int  SearchGreat(
   double  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayDouble::SearchGreat(double) için bir örnek
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchGreat(100.0)!=-1) printf("Eleman bulundu");
   else                             printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLess

Sıralı dizi içinde, belirtilen örnekten daha küçük olan bir eleman arar.

```
int  SearchLess(
   double  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayDouble:: SearchLess(double) için bir örnek
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLess(100.0)!=-1) printf("Eleman bulundu");
   else                            printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchGreatOrEqual

Sıralı dizi içinde, belirtilen örnekten daha büyük veya eşit olan bir eleman arar.

```
int  SearchGreatOrEqual(
   double  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayDouble::SearchGreatOrEqual(double) için bir örnek
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchGreatOrEqual(100.0)!=-1) printf("Eleman bulundu");
   else                                    printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLessOrEqual

Sıralı dizi içinde, belirtilen örnekten daha küçük veya eşit olan bir eleman arar.

```
int  SearchLessOrEqual(
   double  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayDouble::SearchLessOrEqual(double) için bir örnek
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLessOrEqual(100.0)!=-1) printf("Eleman bulundu");
   else                                   printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchFirst

Sıralı dizi içinde, belirtilen kriteri sağlayan ilk elemanı arar.

```
int  SearchFirst(
   double  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayDouble::SearchFirst(double) için bir örnek
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchFirst(100.0)!=-1) printf("Eleman bulundu");
   else                             printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLast

Sıralı dizi içinde, belirtilen kriteri sağlayan son elemanı arar.

```
int  SearchLast(
   double  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayDouble::SearchLast(double) örneği
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLast(100.0)!=-1) printf("Eleman bulundu");
   else                            printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLinear

Belirtilen örneğin sıralı dizideki eşitini arar.

```
int  SearchLinear(
   double  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Not

Yöntem, sıralanmamış diziler için lineer arama (veya ardışık arama) algoritmasını kullanır.

Örnek:

```
//--- CArrayDouble::SearchLinear(double) için bir örnek
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- eleman ara
   if(array.SearchLinear(100.0)!=-1) printf("Eleman bulundu");
   else                              printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

Save

Veri dizisini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayDouble::Save(int) örneği
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- 100 dizi elemanı ekle
   for(int i=0;i<100;i++)
     {
      array.Add(i);
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- dosya kaydetme hatası
         printf("Dosya kaydedilemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

```
      }
   //--- diziyi sil
   delete array;
  }
```

Load

Belirtilen dosyadan veri dizisi yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayDouble::Load(int) örneği
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- dosya yükleme hatası
         printf("Dosya yüklenemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- dizi elemanlarını kullan
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = %f",i,array.At(i));
```

```
      }
   //--- diziyi sil
   delete array;
  }
```

Type

Dizinin tip tanımlayıcısını alır.

```
virtual int  Type() const
```

Dönüş Değeri

Dizinin tip tanımlayıcısı (CArrayDouble için 87).

Örnek:

```
//--- CArrayDouble::Type() için bir örnek
#include <Arrays\ArrayDouble.mqh>
//---
void OnStart()
  {
   CArrayDouble *array=new CArrayDouble;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dizi tipini al
   int type=array.Type();
   //--- diziyi sil
   delete array;
  }
```

CArrayString

CArrayString sınıfı, string tipli değişkenler için bir dinamik dizi sınıfıdır.

Açıklama

CArrayString sınıfı, string tipli değişkenlerden oluşan dinamik dizilerle çalışma olanağı sağlar. Sınıf içerisinde; sıralama, sıralanmış dizide arama, eleman silme/ekleme gibi uygulamalara yer verilmiştir. Ayrıca dosya işlemleri için gereken yöntemler de sınıf içerisinde yer almaktadır.

Bildirim

```
class CArrayString : public CArray
```

Başlık

```
#include <Arrays\ArrayString.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayString

Sınıf Yöntemleri

Bellek denetimi	
Reserve	Dizi boyutunun artırılması için bellek tahsis eder
Resize	Dizi için daha küçük bir boyut ayarlar
Shutdown	Dizi için daha küçük bir boyut ayarlar
Ekleme yöntemleri	
Add	Dizinin sonuna bir eleman ekler
AddArray	Başka bir diziden alınan elemanları dizinin sonuna ekler
AddArray	Başka bir diziden alınan elemanları dizinin sonuna ekler
Insert	Dizinin belirtilen konumuna eleman ekler
InsertArray	Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler
InsertArray	Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler
AssignArray	Dizinin elemanlarını başka bir diziden kopyalar
AssignArray	Dizinin elemanlarını başka bir diziden kopyalar
Güncelleme yöntemleri	
Update	Dizinin belirtilen konumundaki elemanı değiştirir

Bellek denetimi	
Shift	Dizi içindeki bir elemanı belirtilen konuma kaydırır
Silme yöntemleri	
Delete	Dizinin belirtilen konumundaki elemanı siler
DeleteRange	Dizinin belirtilen konumundaki elemanlar grubunu siler
Erişim yöntemleri	
At	Dizinin belirtilen konumundaki elemanı alır
Karşılaştırma yöntemleri	
CompareArray	Diziyi başka bir diziyle karşılaştırır
CompareArray	Diziyi başka bir diziyle karşılaştırır
Sıralı dizi işlemleri	
InsertSort	Sıralı diziye eleman ekler
Search	Sıralı dizi içinde, belirtilen kriteri sağlayan bir eleman arar
SearchGreat	Sıralı dizi içinde, belirtilen örnekten daha büyük olan bir eleman arar
SearchLess	Sıralı dizi içinde, belirtilen örnekten daha küçük olan bir eleman arar
SearchGreatOrEqual	Sıralı dizi içinde, belirtilen örnekten daha büyük veya eşit olan bir eleman arar
SearchLessOrEqual	Sıralı dizi içinde, belirtilen örnekten daha küçük veya eşit olan bir eleman arar
SearchFirst	Sıralı dizi içinde, belirtilen kriteri sağlayan ilk elemanı arar
SearchLast	Sıralı dizi içinde, belirtilen kriteri sağlayan ilk elemanı arar
SearchLinear	Belirtilen örneğin dizi içindeki eşitini arar
Girdi/çıktı	
virtual Save	Veri dizisini dosyaya kaydeder
virtual Load	Veri dizisini dosyadan alır
virtual Type	Dizinin tip tanımlayıcısını alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Reserve

Dizi boyutunun artırılması için bellek tahsis eder.

```
bool  Reserve(
   int  size      // eleman sayısı
   )
```

Parametreler

size

[in] Eklenecek elemanların sayısı.

Dönüş Değeri

Başarılı ise 'true', eleman sayısı sıfır veya daha küçük bir değerle ayarlanmaya çalışılmışsa 'false'.

Not

Bellek bölünmesini azaltmak için, daha önce Step (int) yöntemi ile ayarlanmış olan büyüklük değerini artırın (varsayılan değer 16).

Örnek:

```
//--- CArrayString::Reserve(int) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- bellek tahsis et
   if(!array.Reserve(1024))
     {
      printf("Tahsis hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Resize

Dizi için daha küçük bir boyut ayarlar.

```
bool  Resize(
   int  size      // Boyut
   )
```

Parametreler

size

[in] Yeni dizi boyutu.

Dönüş Değeri

Başarılı ise 'true', boyut değeri sıfır veya daha küçük bir sayıyla ayarlanmaya çalışılmışsa 'false'.

Not

Dizi boyutunun değiştirilmesi belleğin etkin kullanımını sağlar. sağ taraftaki gereksiz elemanlar kaybedilir. Bellek bölünmesini azaltmak için, daha önce Step (int) yöntemi ile ayarlanmış olan büyüklük değerini azaltın (varsayılan değer 16).

Örnek:

```
//--- CArrayString::Resize(int) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi yeniden boyutlandır
   if(!array.Resize(10))
     {
      printf("Boyutlandırma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Shutdown

Belleği boşaltarak diziyi siler.

```
bool  Shutdown()
```

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayString::Shutdown() için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi kapat
   if(!array.Shutdown())
     {
      printf("Kapatma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Add

Dizinin sonuna bir eleman ekler.

```
bool  Add(
   string  element      //    char  element      // eklenecek eleman
   )
```

Parametreler

element

[in] Diziye eklenecek elemanın değeri.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayString::Add(string) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   for(int i=0;i<100;i++)
     {
      if(!array.Add(IntegerToString(i)))
        {
         printf("Eleman ekleme hatası");
         delete array;
         return;
        }
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AddArray

Başka bir diziden alınan elemanları dizinin sonuna ekler.

```
bool  AddArray(
   const string&  src[]      // kaynak dizi
   )
```

Parametreler

src[]

[in] Elemanları eklenecek olan kaynak dizinin referansı.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayString::AddArray(const string &[]) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- yeni dizi ekle
   if(!array.AddArray(src))
     {
      printf("Dizi ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AddArray

Başka bir diziden alınan elemanları dizinin sonuna ekler.

```
bool  AddArray(
   const CArrayString*  src      // kaynağın işaretçisi
   )
```

Parametreler

src

[in] CArrayString sınıf örneğinin (eklenecek kaynak elemanlar) işaretçisi.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayString::AddArray(const CArrayString*) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- yeni dizi ekle
   if(!array.AddArray(src))
     {
      printf("Dizi ekleme hatası");
      delete src;
      delete array;
      return;
     }
   //--- kaynak diziyi sil
   delete src;
```

```
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Insert

Dizinin belirtilen konumuna eleman ekler.

```
bool  Insert(
   string  element,     // eklenecek eleman
   int     pos          // konum
   )
```

Parametreler

element

[in] Diziye eklenecek elemanın değeri

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayString::Insert(string,int) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- eleman ekle
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(IntegerToString(i),0))
        {
         printf("Ekleme hatası");
         delete array;
         return;
        }
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

InsertArray

Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler.

```
bool  InsertArray(
   const string&  src[],      // kaynak dizi
   int             pos         // konum
   )
```

Parametreler

src[]

[in] Elemanları eklenecek olan kaynak dizinin referansı

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', elemanlar eklenemezse 'false'.

Örnek:

```
//--- CArrayString::InsertArray(const string &[],int) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ekle
   if(!array.InsertArray(src,0))
     {
      printf("dizi ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

InsertArray

Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler.

```
bool  InsertArray(
   CArrayString*  src,      // kaynağın işaretçisi
   int             pos      // konum
   )
```

Parametreler

src

[in] CArrayString sınıf örneğinin (eklenecek kaynak elemanlar) işaretçisi.

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', elemanlar eklenemezse 'false'.

Örnek:

```
//--- CArrayString::InsertArray(const CArrayString*,int) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ekle
   if(!array.InsertArray(src,0))
     {
      printf("dizi ekleme hatası");
      delete src;
      delete array;
```

```
      return;
     }
   //--- kaynak diziyi sil
   delete src;
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AssignArray

Dizinin elemanlarını başka bir diziden kopyalar

```
bool  AssignArray(
   const string&  src[]      // kaynak dizi
   )
```

Parametreler

src[]

[in] Kaynak değerleri içeren dizinin referansı.

Dönüş Değeri

Başarılı ise 'true', elemanlar kopyalanamazsa 'false'.

Örnek:

```
//--- CArrayString::AssignArray(const string &[]) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ata
   if(!array.AssignArray(src))
     {
      printf("Dizi atama hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AssignArray

Dizinin elemanlarını başka bir diziden kopyalar

```
bool  AssignArray(
   const CArrayString*  src      // kaynağın işaretçisi
   )
```

Parametreler

src

[in] CArrayString sınıf örneğinin (kopyalanacak kaynak elemanlar) işaretçisi.

Dönüş Değeri

Başarılı ise 'true', elemanlar kopyalanamazsa 'false'.

Örnek:

```
//--- CArrayString::AssignArray(const CArrayString*) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayString *src  =new CArrayString;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ata
   if(!array.AssignArray(src))
     {
      printf("Dizi atama hatası");
      delete src;
      delete array;
      return;
     }
   //--- diziler aynı
   //--- kaynak diziyi sil
```

```
   delete src;
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Update

Dizinin belirtilen konumundaki elemanı değiştirir.

```
bool  Update(
   int     pos,          // konum
   string  element       // değer
   )
```

Parametreler

pos

[in] Değiştirilecek elemanın dizideki konumu

element

[in] Yeni elemanın değeri

Dönüş Değeri

Başarılı ise 'true', eleman değiştirilemezse 'false'.

Örnek:

```
//--- CArrayString::Update(int, string) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı güncelle
   if(!array.Update(0,"ABC"))
     {
      printf("Güncelleme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Shift

Dizi içindeki bir elemanı belirtilen konuma kaydırır.

```
bool  Shift(
   int  pos,        // konum
   int  shift       // Kaydırma değeri
   )
```

Parametreler

pos

[in] Kaydırılacak dizi elemanının konumu

shift

[in] Kaydırma değeri (pozitif veya negatif).

Dönüş Değeri

Başarılı ise 'true', eleman kaydırılamadıysa 'false'.

Örnek:

```
//--- CArrayString::Shift(int,int) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı kaydır
   if(!array.Shift(10,-5))
     {
      printf("Kaydırma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Delete

Dizinin belirtilen konumundaki elemanı siler.

```
bool  Delete(
   int  pos      // konum
   )
```

Parametreler

pos

[in] Silinecek elemanın konumu.

Dönüş Değeri

Başarılı ise 'true', eleman silinemezse 'false'.

Örnek:

```
//--- CArrayString::Delete(int) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı sil
   if(!array.Delete(0))
     {
      printf("Silme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

DeleteRange

Dizinin belirtilen konumundaki elemanlar grubunu siler.

```
bool  DeleteRange(
   int  from,     // ilk elemanın konumu
   int  to        // son elemanın konumu
   )
```

Parametreler

from

[in] Silinecek ilk elemanın konumu.

to

[in] Silinecek son elemanın konumu.

Dönüş Değeri

Başarılı ise 'true', elemanlar silinemezse 'false'.

Örnek:

```
//--- CArrayString::DeleteRange(int,int) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanları sil
   if(!array.DeleteRange(0,10))
     {
      printf("Silme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

At

Dizinin belirtilen konumundaki elemanı alır.

```
string  At(
   int  pos      // konum
   ) const
```

Parametreler

pos

[in] Seçilen elemanın dizi içindeki konumu.

Dönüş Değeri

Başarı durumunda elemanın değerine, olmayan bir konumdaki bir elemanın istenmesi durumunda ise "" değerine dönüş yapar (son hata değeri ERR_OUT_OF_RANGE).

Not

"" değeri geçerli bir dizi elemanı olabileceği için her zaman son hata kodunun istenmesi gerekir.

Örnek:

```
//--- CArrayString::At(int) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      string result=array.At(i);
      if(result=="" && GetLastError()==ERR_OUT_OF_RANGE)
        {
         //--- Dizi okuma hatası
         printf("Eleman alınamadı, hata");
         delete array;
         return;
        }
      //--- elemanı kullan
      //--- . . .
     }
   //--- diziyi sil
```

```
   delete array;
  }
```

CompareArray

Diziyi başka bir diziyle karşılaştırır.

```
bool  CompareArray(
   const string&  src[]      // kaynak dizi
   ) const
```

Parametreler

src[]

[in] Karşılaştırma için kullanılacak dizinin referansı.

Dönüş Değeri

Diziler eşit ise 'true', aksi durumda 'false'

Örnek:

```
//--- CArrayString::CompareArray(const string &[]) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
string src[];
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- başka bir dizi ile karşılaştır
   int result=array.CompareArray(src);
   //--- diziyi sil
   delete array;
  }
```

CompareArray

Diziyi başka bir diziyle karşılaştırır.

```
bool  CompareArrays(
   const CArrayString*  src      // kaynağın işaretçisi
   ) const
```

Parametreler

src

[in] CArrayString sınıf örneğinin işaretçisi (karşılaştırma için kaynak elemanlar).

Dönüş Değeri

Diziler eşit ise 'true', aksi durumda 'false'

Örnek:

```
//--- CArrayString::CompareArray(const CArrayString*) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayString *src=new CArrayString;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- kaynak dizinin elemanlarını ekle
   //--- . . .
   //--- başka bir dizi ile karşılaştır
   int result=array.CompareArray(src);
   //--- dizileri sil
   delete src;
   delete array;
  }
```

InsertSort

Sıralı diziye eleman ekler.

```
bool  InsertSort(
   string  element      // eklenecek eleman
   )
```

Parametreler

element

[in] Sıralı diziye eklenecek elemanın değeri

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Örnek:

```
//--- CArrayString::InsertSort(string) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ekle
   if(!array.InsertSort("ABC"))
     {
      printf("Ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Search

Belirtilen örneğin sıralı dizideki eşitini arar.

```
int  Search(
   string  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayString::Search(string) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.Search("ABC")!=-1) printf("Eleman bulundu");
   else                        printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchGreat

Sıralı dizi içinde, belirtilen örnekten daha büyük olan bir eleman arar.

```
int  SearchGreat(
   string  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayString::SearchGreat(string) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchGreat("ABC")!=-1) printf("Eleman bulundu");
   else                             printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLess

Sıralı dizi içinde, belirtilen örnekten daha küçük olan bir eleman arar.

```
int  SearchLess(
   string  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayString:: SearchLess(string) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLess("ABC")!=-1) printf("Eleman bulundu");
   else                            printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchGreatOrEqual

Sıralı dizi içinde, belirtilen örnekten daha büyük veya eşit olan bir eleman arar.

```
int  SearchGreatOrEqual(
   string  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayString:: SearchGreatOrEqual(string) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchGreatOrEqual("ABC")!=-1) printf("Eleman bulundu");
   else                                    printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLessOrEqual

Sıralı dizi içinde, belirtilen örnekten daha küçük veya eşit olan bir eleman arar.

```
int  SearchLessOrEqual(
   string  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//---CArrayString:: SearchLessOrEqual(string) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLessOrEqual("ABC")!=-1) printf("Eleman bulundu");
   else                                   printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchFirst

Sıralı dizi içinde, belirtilen kriteri sağlayan ilk elemanı arar.

```
int  SearchFirst(
   string  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayString:: SearchFirst(string) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchFirst("ABC")!=-1) printf("Eleman bulundu");
   else                             printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLast

Sıralı dizi içinde, belirtilen kriteri sağlayan son elemanı arar.

```
int  SearchLast(
   string  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayString:: SearchLast(string) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ara
   if(array.SearchLast("ABC")!=-1) printf("Eleman bulundu");
   else                            printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLinear

Belirtilen örneğin dizi içindeki eşitini arar.

```
int  SearchLinear(
   string  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Not

Yöntem, sıralanmamış diziler için lineer arama (veya ardışık arama) algoritmasını kullanır.

Örnek:

```
//--- CArrayString::SearchLinear(string) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- eleman ara
   if(array.SearchLinear("ABC")!=-1) printf("Eleman bulundu");
   else                              printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

Save

Veri dizisini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayString::Save(int) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayString *array=new CArrayString;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- 100 dizi elemanı ekle
   for(int i=0;i<100;i++)
     {
      array.Add(IntegerToString(i));
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- dosya kaydetme hatası
         printf("Dosya kaydedilemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
```

```
      }
   delete array;
  }
```

Load

Belirtilen dosyadan veri dizisi yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayString::Load(int) için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CArrayString *array=new CArrayString;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- dosya yükleme hatası
         printf("Dosya yüklenemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- dizi elemanlarını kullan
   for(int i=0;i<array.Total();i++)
     {
      printf("Element[%d] = '%s'",i,array.At(i));
```

```
      }
   delete array;
  }
```

Type

Dizinin tip tanımlayıcısını alır.

```
virtual int  Type() const
```

Dönüş Değeri

Dizinin tip tanımlayıcısı (CArrayString için 89).

Örnek:

```
//--- CArrayString::Type() için bir örnek
#include <Arrays\ArrayString.mqh>
//---
void OnStart()
  {
   CArrayString *array=new CArrayString;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dizi tipini al
   int type=array.Type();
   //--- diziyi sil
   delete array;
  }
```

CArrayObj

CArrayObj sınıfı, CObject ve onun soyundan gelen sınıf örneklerinin işaretçileri için bir dinamik dizi sınıfıdır.

Açıklama

CArrayObj sınıfı, CObject ve onun soyundan gelen sınıfların işaretçilerinden oluşan dinamik dizilerle çalışmaya olanak sağlar. Yani, basit tiplerin çok boyutlu dinamik dizileriyle ve daha karmaşık organize veri yapılarıyla çalışma imkanı sağlar.

Sınıf içerisinde; sıralama, sıralanmış dizide arama, eleman silme/ekleme gibi işlevler yer almaktadır. Ayrıca dosya işlemleri için gereken yöntemler de sınıf içerisinde yer almaktadır.

CArrayObj sınıfı ile çalışmak bir takım incelikler içerir.

Bildirim

```
class CArrayObj : public CArray
```

Başlık

```
#include <Arrays\ArrayObj.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj

İlk nesil

CIndicators, CSeries

Class Method

Özellikler	
FreeMode	Bellek yönetim bayrağını alır
FreeMode	Bellek yönetim bayrağını ayarlar
Bellek denetimi	
Reserve	Dizi boyutunun artırılması için bellek tahsis eder
Resize	Dizi için daha küçük bir boyut ayarlar
Shutdown	Belleği boşaltarak diziyi siler.
Ekleme yöntemleri	
Add	Dizinin sonuna bir eleman ekler
AddArray	Dizinin sonuna bir eleman ekler

Özellikler	
Insert	Dizinin belirtilen konumuna eleman ekler
InsertArray	Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler
AssignArray	Dizinin elemanlarını başka bir diziden kopyalar
Güncelleme yöntemleri	
Update	Dizinin belirtilen konumundaki elemanı değiştirir
Shift	Dizi içindeki bir elemanı belirtilen konuma kaydırır
Silme yöntemleri	
Detach	Dizinin belirtilen konumundaki elemanı alır ve onu diziden kaldırır
Delete	Dizinin belirtilen konumundaki elemanı siler
DeleteRange	Dizinin belirtilen konumundaki elemanlar grubunu siler
Clear	Belleği boşaltmadan dizi elemanlarının tamamını temizler
Erişim yöntemleri	
At	Dizinin belirtilen konumundaki elemanı alır
Karşılaştırma yöntemleri	
CompareArray	Diziyi başka bir diziyle karşılaştırır
Sıralı dizi işlemleri	
InsertSort	Sıralı diziye eleman ekler
Search	Sıralı dizi içinde, belirtilen örneğe eşit olan bir eleman arar
SearchGreat	Sıralı dizi içinde, belirtilen örnekten daha büyük olan bir eleman arar
SearchLess	Sıralı dizi içinde, belirtilen örnekten daha küçük olan bir eleman arar
SearchGreatOrEqual	Sıralı dizi içinde, belirtilen örnekten daha büyük veya eşit olan bir eleman arar
SearchLessOrEqual	Sıralı dizi içinde, belirtilen örnekten daha küçük veya eşit olan bir eleman arar
SearchFirst	Sıralı dizi içinde, belirtilen kriteri sağlayan ilk elemanı arar
SearchLast	Sıralı dizi içinde, belirtilen kriteri sağlayan son elemanı arar
Girdi/çıktı	
Save	Veri dizisini dosyaya kaydeder
Load	Veri dizisini dosyadan alır
Type	Dizinin tip tanımlayıcısını alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Dizilere dair pratik uygulamaların tamamı CObject sınıfının soyundan gelir (standart kütüphanenin tüm sınıfları da buna dahildir).

Örneğin, iki boyutlu diziler için seçeneklere bakalım:

```
#include <Arrays\ArrayDouble.mqh>
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int i,j;
   int first_size=10;
   int second_size=100;
//--- diziyi oluştur
   CArrayObj    *array=new CArrayObj;
   CArrayDouble *sub_array;
//---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
//--- alt diziler oluştur
   for(i=0;i<first_size;i++)
     {
      sub_array=new CArrayDouble;
      if(sub_array==NULL)
        {
         delete array;
         printf("Nesne oluşturma hatası");
         return;
        }
      //--- diziyi doldur
      for(j=0;j<second_size;j++)
        {
         sub_array.Add(i*j);
        }
      array.Add(sub_array);
     }
//--- dizi oluşturma tamam
   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);
         //--- dizi elemanını kullan
        }
      }
   delete array;
  }
```

İnce işçilik

Bu sınıf dinamik belleği denetlemek için mekanizmalara sahiptir, bu nedenle dizi elemanlarıyla çalışılırken dikkatli olunması gerekir.

Bellek yönetimi mekanizması FreeMode (bool) yöntemi kullanılarak açılıp kapatılabilir. Bu mekanizme varsayılan olarak etkin durumdadır.

Dolayısıyla, CArrayObj sınıfıyla çalışmak için iki seçenek mevcuttur:

1. Bellek yönetimi mekanizması devrede. (varsayılan)

Bu seçenekte CArrayObj, elemanların diziden kaldırılmasının ardından belleği boşaltma sorumluluğunu alır. Bu durumda kullanıcı dizi elemanlarını el yordamıyla boşaltmamalıdır.

Örnek:

```
   int i;
//--- Bir dizi oluştur
   CArrayObj *array=new CArrayObj;
//--- Diziyi doldur
   for(i=0;i<10;i++) array.Add(new CObject);
//--- Birşeyler yap
   for(i=0;i<array.Total();i++)
     {
      CObject *object=array.At(i);
      //--- bir elemanla işlem
      . . .
     }
//--- Diziyi elemanlarıyla birlikte kaldır
   delete array;
```

2. Bellek yönetimi mekanizması devre-dışı.

Bu seçenekte CArrayObj, elemanların diziden kaldırılmasının ardından belleği boşaltma sorumluluğunu almayacaktır. Dolayısıyla, dizi elemanları kullanıcı tarafından serbest bırakılmalıdır.

Örnek:

```
   int i;
//--- Bir dizi oluştur
   CArrayObj *array=new CArrayObj;
```

```
//--- Bellek yönetimi mekanizmasını devre-dışı bırak
   array.FreeMode(false);
//--- Diziyi doldur
   for(i=0;i<10;i++) array.Add(new CObject);
//--- Birşeyler yap
   for(i=0;i<array.Total();i++)
     {
      CObject *object=array.At(i);
      //--- bir elemanla işlem
      . . .
     }
//--- Dizi elemanlarını kaldır
   while(array.Total()) delete array.Detach();
//--- Boş diziyi sil
   delete array;
```

FreeMode

Bellek yönetimi bayrağını alır.

```
bool  FreeMode() const
```

Dönüş Değeri

Bellek yönetimi bayrağı.

Örnek:

```
//--- CArrayObj::FreeMode() için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- serbest durum bayrağını al
   bool array_free_mode=array.FreeMode();
   //--- diziyi sil
   delete array;
  }
```

FreeMode

Bellek yönetim bayrağını ayarlar.

```
void  FreeMode(
   bool  mode      // yeni bayrak
   )
```

Parametreler

mode

[in] Bellek yönetim bayrağı için yeni değer.

Dönüş Değeri

Yok.

Not

Bellek yönetimi bayrağının ayarlanması CArrayObj kullanımında önemli bir noktadır. Dizi elemanları dinamik nesne işaretçileri olduğundan, diziden kaldırıldıklarında akıbetlerinin ne olacağını belirlemek oldukça önemlidir.

Bayrak ayarlanmışsa, elemanın diziden kaldırılmasının ardından 'delete' operatörü ile eleman otomatik olarak silinecektir. Bayrak ayarlanmamışsa, silinen elemanın işaretçisinin hala program içerisinde bir yerlerde olduğu varsayılır ve daha sonra programdan kaldırılacaktır.

Kullanıcının bellek yönetim bayrağını sıfırlaması durumunda, program tamamlanmadan dizinin kaldırılması gereklidir, aksi takdirde, 'new' operatörü kullanıldığında bellek boşaltılmamış elemanlarla dolacaktır.

Büyük veri miktarları terminalin eninde sonunda çökmesine hatta bozulmasına yol açacaktır. Kullanıcının bellek yönetim bayrağını sıfırlamaması durumunda ise başka bir "tehlike" doğacaktır.

İşaretçilerin kullanılması durumunda dizi elemanları yerel değişkenler içinde depolanacaktır. Bu durumda dizinin kaldırılması kritik hataya yol açacak ve programı çökertecektir. Varsayılan olarak, bellek yönetim bayrağı ayarlıdır (yönetim mekanizması açık), yani belleğin boşaltılmasından dizi sınıfı sorumludur.

Örnek:

```
//--- CArrayObj::FreeMode(bool) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- serbest durum bayrağını sıfırla
```

```
   array.FreeMode(false);
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Reserve

Dizi boyutunun artırılması için bellek tahsis eder.

```
bool  Reserve(
   int  size      // eleman sayısı
   )
```

Parametreler

size

[in] Eklenecek elemanların sayısı.

Dönüş Değeri

Başarılı ise 'true', eleman sayısı sıfır veya daha küçük bir değerle ayarlanmaya çalışılmışsa 'false'.

Not

Bellek bölünmesini azaltmak için, daha önce Step (int) yöntemi ile ayarlanmış olan büyüklük değerini artırın (varsayılan değer 16).

Örnek:

```
//--- CArrayObj::Reserve(int) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   if(!array.Reserve(1024))
     {
      printf("Tahsis hatası");
      delete array;
      return;
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Resize

Dizi için daha küçük bir boyut ayarlar.

```
bool  Resize(
   int  size      // Boyut
   )
```

Parametreler

size

[in] Yeni dizi boyutu.

Dönüş Değeri

Bellek bölünmesini azaltmak için, daha önce Step (int) yöntemi ile ayarlanmış olan büyüklük değerini değiştirin (varsayılan değer 16).

Not

Dizinin boyutunu değiştirmek belleğin optimal kullanımını sağlayabilir. Sağ tarafa yüklenmiş aşırı elemanlar kaybedilir. Kaybolan elemanlar için tahsis edilmiş belleğin temizlenmesi, bellek yönetimi mekanizmasının durumuna bağlıdır.

Bellek bölünmesini azaltmak için, daha önce Step (int) yöntemi ile ayarlanmış olan büyüklük değerini değiştirin (varsayılan değer 16).

Örnek:

```
//--- CArrayObj::Resize(int) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi yeniden boyutlandır
   if(!array.Resize(10))
     {
      printf("Boyutlandırma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
```

```
}
```

Clear

Belleği boşaltmadan dizi elemanlarının tamamını temizler.

```
void  Clear()
```

Dönüş Değeri

Yok.

Not

Bellek yönetimi mekanizması açıksa, silinene elemanlar bellekten boşaltılır.

Örnek:

```
//--- CArrayObj::Clear() için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi temizle
   array.Clear();
   //--- diziyi sil
   delete array;
  }
```

Shutdown

Belleği boşaltarak diziyi siler.

```
bool  Shutdown()
```

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Bellek yönetimi mekanizması açıksa, silinene elemanlar bellekten boşaltılır.

Örnek:

```
//--- CArrayObj::Shutdown() için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi kapat
   if(!array.Shutdown())
     {
      printf("Kapatma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

CreateElement

Dizinin belirtilen konumunda yeni eleman oluşturur.

```
bool  CreateElement(
   int  index      // konum
   )
```

Parametreler

index

[in] Yeni elemanın oluşturulacağı dizi konumu.

Dönüş Değeri

Başarılı ise 'true', eleman oluşturulamazsa 'false'.

Not

CArrayObj sınıfındaki CreateElement (int) yöntemi her zaman 'false' dönüşü yapacak ve herhangi bir eylem gerçekleştirmeyecektir. Gerekli görüldüğünde, CreateElement (int) yöntemi türetik sınıf içinde uygulanmalıdır.

Örnek:

```
//--- CArrayObj::CreateElement(int) için bir örnek.
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int         size=100;
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziyi doldur
   array.Reserve(size);
   for(int i=0;i<size;i++)
     {
      if(!array.CreateElement(i))
        {
         printf("Eleman oluşturma hatası");
         delete array;
         return;
        }
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
```

```
   delete array;
  }
```

Add

Dizinin sonuna bir eleman ekler.

```
bool  Add(
   CObject*  element      // eklenecek eleman
   )
```

Parametreler

element

[in] Diziye eklenecek elemanın değeri.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Not

Parametrenin hatalı işaretçi ile geçirilmesi (NULL gibi) durumunda, eleman diziye eklenmeyecektir.

Örnek:

```
//--- CArrayObj::Add(CObject*) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- 100 dizi elemanı ekle
   for(int i=0;i<100;i++)
     {
      if(!array.Add(new CObject))
        {
         printf("Eleman ekleme hatası");
         delete array;
         return;
        }
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AddArray

Başka bir diziden alınan elemanları dizinin sonuna ekler.

```
bool  AddArray(
   const CArrayObj *  src      // kaynak dizinin işaretçisi
   )
```

Parametreler

src

[in] CArrayDouble sınıf örneğinin işaretçisi - eklenecek elemanların kaynağı.

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Not

Dizi elemanlarının diziye eklenmesi aslında işaretçilerin kopyalanması anlamına gelir. Bu nedenle yöntemin çağrılması bir "tehlike" yaratabilir — bir dinamik nesnenin işaretçisi birden fazla değişken içerisinde olabilir.

```
//--- örnek
extern bool       make_error;
extern int        error;
extern CArrayObj *src;
//--- Yeni bir CArrayObj örneği oluştur
//--- Varsayılan olarak bellek yönetimi açık
CArrayObj *array=new CArrayObj;
//--- kaynak dizinin elemanlarını ekle (kopyala)
if(array!=NULL)
   bool result=array.AddArray(src);
if(make_error)
  {
   //--- hatalı işlemler yap
   switch(error)
     {
      case 0:
         //--- bellek yönetim bayrağını denetlemeden kaynak diziyi kaldır
         delete src;
         //--- Sonuç:
         //--- bir elemanı, alıcı dizideki geçersiz bir işaretçiyle adreslemek mümkünc
         break;
      case 1:
         //--- kaynak dizideki bellek yönetim mekanizmasını devre dışı bırak
         if(src.FreeMode()) src.FreeMode(false);
         //--- Ama kaynak diziyi silme
         //--- Sonuç:
         //--- kaldırma işleminin ardından bir elemanı, kaynak dizideki geçersiz bir i
         break;
```

```
      case 2:
         //--- kaynak dizideki bellek yönetim mekanizmasını devre dışı bırak
         src.FreeMode(false);
         //--- alıcı dizideki bellek yönetim mekanizmasını devre-dışı bırak
         array.FreeMode(false);
         //--- Sonuç:
         //--- Programın sonlandırılmasının ardından bir "bellek sızıntısı" al
         break;
      }
  }
else
  {
   //--- kaynak dizideki bellek yönetim mekanizmasını devre dışı bırak
   if(src.FreeMode()) src.FreeMode(false);
   //--- kaynak diziyi sil
   delete src;
   //--- Sonuç:
   //--- alıcı dizi elemanının adreslenmesi doğru olacak
   //--- Alıcı dizinin silinmesi elemanlarının da silinmesine yol açacak
  }
```

Örnek:

```
//--- CArrayObj::AddArray(const CArrayObj*)için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- serbest durum bayrağını sıfırla
   src.FreeMode(false);
   //--- diziyi doldur
   //--- . . .
   //--- yeni dizi ekle
```

```
   if(!array.AddArray(src))
     {
      printf("Dizi ekleme hatası");
      delete src;
      delete array;
      return;
     }
   //--- kaynak diziyi elemanlarını silmeden kaldır
   delete src;
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Insert

Dizinin belirtilen konumuna eleman ekler.

```
bool  Insert(
   CObject*  element,      // eklenecek eleman
   int       pos           // konum
   )
```

Parametreler

element

[in] Diziye eklenecek elemanın değeri

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Not

Parametrenin hatalı işaretçi ile geçirilmesi (NULL gibi) durumunda, eleman diziye eklenmeyecektir.

Örnek:

```
//--- CArrayObj::Insert(CObject*,int) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- eleman ekle
   for(int i=0;i<100;i++)
     {
      if(!array.Insert(new CObject,0))
        {
         printf("Ekleme hatası");
         delete array;
         return;
        }
     }
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
```

```
   delete array;
  }
```

InsertArray

Başka bir diziden alınan elemanları dizinin belirtilen konumuna ekler.

```
bool  InsertArray(
   const CArrayObj*  src,      // kaynak işaretçisi
   int               pos       // konum
   )
```

Parametreler

src

[in] CArrayObj sınıf örneğinin —eklenecek elemanların işaretçisi.

pos

[in] Elemanın ekleneceği dizi konumu

Dönüş Değeri

Başarılı ise 'true', elemanlar eklenemezse 'false'.

Not

See: CArrayObj::AddArray(const CArrayObj*).

Örnek:

```
//--- CArrayObj::InsertArray(const CArrayObj*,int) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- serbest durum bayrağını sıfırla
   src.FreeMode(false);
   //--- diziyi doldur
   //--- . . .
   //--- başka bir dizi ekle
```

```
   if(!array.InsertArray(src,0))
     {
      printf("dizi ekleme hatası");
      delete src;
      delete array;
      return;
     }
   //--- kaynak diziyi elemanlarını silmeden kaldır
   delete src;
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

AssignArray

Dizinin elemanlarını başka bir diziden kopyalar

```
bool  AssignArray(
   const CArrayObj*  src      // kaynağın işaretçisi
   )
```

Parametreler

src

[in] CArrayObj sınıf örneğinin (kopyalanacak kaynak elemanlar) işaretçisi.

Dönüş Değeri

Başarılı ise 'true', elemanlar kopyalanamazsa 'false'.

Not

Alıcı dizi AssignArray çağrısından önce boş değilse tüm elemanları diziden kaldırılır ve bellek yönetimi açıksa silinen elemanlar serbest bırakılır. Bu durumda alıcı dizi kaynak dizinin kopyası olur. Ayrıca bakın: CArrayObj::AddArray(const CArrayObj*).

Örnek:

```
//--- CArrayObj::AssignArray(const CArrayObj*) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- serbest durum bayrağını sıfırla
   src.FreeMode(false);
   //--- diziyi doldur
   //--- . . .
   //--- başka bir dizi ata
   if(!array.AssignArray(src))
     {
```

```
      printf("Dizi atama hatası");
      delete src;
      delete array;
      return;
     }
   //--- diziler aynı
   //--- kaynak diziyi elemanlarını silmeden kaldır
   delete src;
   //--- diziyi kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Update

Dizinin belirtilen konumundaki elemanı değiştirir.

```
bool  Update(
   int       pos,          // konum
   CObject*  element       // değer
   )
```

Parametreler

pos

[in] Değiştirilecek elemanın dizideki konumu

element

[in] Yeni elemanın değeri

Dönüş Değeri

Başarılı ise 'true', eleman değiştirilemezse 'false'.

Not

Geçersiz bir işaretçinin (NULL gibi) parametre olarak geçirilmesi durumunda eleman değişmeyecektir. Bellek yönetimi etkinse kalıntı bellek boşaltılır.

Örnek:

```
//--- CArrayObj::Update(int,CObject*) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı güncelle
   if(!array.Update(0,new CObject))
     {
      printf("Güncelleme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Shift

Dizi içindeki bir elemanı belirtilen konuma kaydırır.

```
bool  Shift(
   int  pos,        // konum
   int  shift       // Kaydırma değeri
   )
```

Parametreler

pos

[in] Kaydırılacak dizi elemanının konumu

shift

[in] Kaydırma değeri (pozitif veya negatif).

Dönüş Değeri

Başarılı ise 'true', eleman kaydırılamadıysa 'false'.

Örnek:

```
//--- CArrayObj::Shift(int,int) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanı kaydır
   if(!array.Shift(10,-5))
     {
      printf("Kaydırma hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Detach

Dizinin belirtilen konumundaki elemanı diziden ayırır.

```
CObject*  Detach(
   int  pos      // konum
   )
```

Parametreler

pos

[in] Seçilen elemanın dizi içindeki konumu.

Dönüş Değeri

Kaldırılan elemanın işaretçisi. Eleman kaldırılamamışsa NULL dönüşü yapar.

Not

Eleman diziden ayrıldığında, bellek yönetim bayrağının herhangi bir durumu için bellekten kaldırılmaz. Ayrılan elemanın işaretçisi kullanımdan sonra bellekten silinmelidir.

Örnek:

```
//--- CArrayObj::Detach(int) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   CObject *object=array.Detach(0);
   if(object==NULL)
     {
      printf("Ayırma hatası");
      delete array;
      return;
     }
   //--- elemanı kullan
   //--- . . .
   //--- elemanı sil
   delete object;
   //--- diziyi sil
   delete array;
  }
```

Delete

Dizinin belirtilen konumundaki elemanı siler.

```
bool  Delete(
   int  pos      // konum
   )
```

Parametreler

pos

[in] Silinecek elemanın konumu.

Dönüş Değeri

Başarılı ise 'true', eleman silinemezse 'false'.

Not

Bellek yönetimi mekanizması açıksa, silinene elemanlar bellekten boşaltılır.

Örnek:

```
//--- CArrayObj::Delete(int) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   if(!array.Delete(0))
     {
      printf("Silme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

DeleteRange

Dizinin belirtilen konumundaki elemanlar grubunu siler.

```
bool  DeleteRange(
   int  from,     // ilk elemanın konumu
   int  to        // son elemanın konumu
   )
```

Parametreler

from

[in] Silinecek ilk elemanın konumu.

to

[in] Silinecek son elemanın konumu.

Dönüş Değeri

Başarılı ise 'true', elemanlar silinemezse 'false'.

Not

Bellek yönetimi mekanizması açıksa, silinene elemanlar bellekten boşaltılır.

Örnek:

```
//--- CArrayObj::DeleteRange(int,int) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- elemanları sil
   if(!array.DeleteRange(0,10))
     {
      printf("Silme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

At

Dizinin belirtilen konumundaki elemanı alır.

```
CObject*  At(
   int  pos      // konum
   )
```

Parametreler

pos

[in] Seçilen elemanın dizi içindeki konumu.

Dönüş Değeri

Başarı durumunda elemanın değerine, olmayan bir konumdaki bir elemanın istenmesi durumunda ise NULL değerine dönüş yapar.

Örnek:

```
//--- CArrayObj::At(int) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- eleman ekle
   //--- . . .
   for(int i=0;i<array.Total();i++)
     {
      CObject *result=array.At(i);
      if(result==NULL)
        {
         //--- Dizi okuma hatası
         printf("Eleman alınamadı, hata");
         delete array;
         return;
        }
      //--- elemanı kullan
      //--- . . .
     }
   delete array;
  }
```

CompareArray

Diziyi başka bir diziyle karşılaştırır.

```
bool  CompareArray(
   const CArrayObj*  src      // kaynağın işaretçisi
   ) const
```

Parametreler

src

[in] CArrayObj sınıf örneğinin işaretçisi - karşılaştırma yapılacak kaynak elemanlar.

Dönüş Değeri

Diziler eşit ise 'true', aksi durumda 'false'

Örnek:

```
//--- CArrayObj::CompareArray(const CArrayObj*) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak diziyi oluştur
   CArrayObj *src=new CArrayObj;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete array;
      return;
     }
   //--- diziyi doldur
   //--- . . .
   //--- başka bir dizi ile karşılaştır
   int result=array.CompareArray(src);
   //--- dizileri sil
   delete src;
   delete array;
  }
```

InsertSort

Sıralı diziye eleman ekler.

```
bool  InsertSort(
   CObject*  element      // eklenecek eleman
   )
```

Parametreler

element

[in] Sıralı diziye eklenecek elemanın değeri

Dönüş Değeri

Başarılı ise 'true', eleman eklenemezse 'false'.

Not

Parametrenin hatalı işaretçi ile geçirilmesi (NULL gibi) durumunda, eleman diziye eklenmeyecektir.

Örnek:

```
//--- CArrayObj::InsertSort(CObject*) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- eleman ekle
   if(!array.InsertSort(new CObject))
     {
      printf("Ekleme hatası");
      delete array;
      return;
     }
   //--- diziyi sil
   delete array;
  }
```

Search

Belirtilen örneğin sıralı dizideki eşitini arar.

```
int  Search(
   CObject*  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayObj::Search(CObject*) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- örneği oluştur
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Örnek oluşturma hatası");
      delete array;
      return;
     }
   //--- örnek özelliklerini ayarla
   //--- . . .
   //--- eleman ara
   if(array.Search(sample)!=-1) printf("Eleman bulundu");
   else                         printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchGreat

Sıralı dizi içinde, belirtilen örnekten daha büyük olan bir eleman arar.

```
int  SearchGreat(
   CObject*  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayObj::SearchGreat(CObject*) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- örneği oluştur
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Örnek oluşturma hatası");
      delete array;
      return;
     }
   //--- örnek özelliklerini ayarla
   //--- . . .
   //--- eleman ara
   if(array.SearchGreat(sample)!=-1) printf("Eleman bulundu");
   else                              printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLess

Sıralı dizi içinde, belirtilen örnekten daha küçük olan bir eleman arar.

```
int  SearchLess(
   CObject*  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayObj:: SearchLess(CObject*) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- örneği oluştur
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Örnek oluşturma hatası");
      delete array;
      return;
     }
   //--- örnek özelliklerini ayarla
   //--- . . .
   //--- eleman ara
   if(array.SearchLess(sample)!=-1) printf("Eleman bulundu");
   else                             printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchGreatOrEqual

Sıralı dizi içinde, belirtilen örnekten daha büyük veya eşit olan bir eleman arar.

```
int  SearchGreatOrEqual(
   CObject*  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayObj::SearchGreatOrEqual(CObject*) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- örneği oluştur
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Örnek oluşturma hatası");
      delete array;
      return;
     }
   //--- örnek özelliklerini ayarla
   //--- . . .
   //--- eleman ara
   if(array.SearchGreatOrEqual(sample)!=-1) printf("Eleman bulundu");
   else                                     printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLessOrEqual

Sıralı dizi içinde, belirtilen örnekten daha küçük veya eşit olan bir eleman arar.

```
int  SearchLessOrEqual(
   CObject*  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayObj:: SearchLessOrEqual(CObject*) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- örneği oluştur
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Örnek oluşturma hatası");
      delete array;
      return;
     }
   //--- örnek özelliklerini ayarla
   //--- . . .
   //--- eleman ara
   if(array.SearchLessOrEqual(sample)!=-1) printf("Eleman bulundu");
   else                                    printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchFirst

Sıralı dizi içinde, belirtilen kriteri sağlayan ilk elemanı arar.

```
int  SearchFirst(
   CObject*  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayObj::SearchFirst(CObject*) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- örneği oluştur
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Örnek oluşturma hatası");
      delete array;
      return;
     }
   //--- örnek özelliklerini ayarla
   //--- . . .
   //--- eleman ara
   if(array.SearchFirst(sample)!=-1) printf("Eleman bulundu");
   else                              printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

SearchLast

Sıralı dizi içinde, belirtilen kriteri sağlayan son elemanı arar.

```
int  SearchLast(
   CObject*  element      // örnek
   ) const
```

Parametreler

element

[in] dizi içinde aranacak örnek eleman.

Dönüş Değeri

Başarı durumunda bulunan elemanın konumu, eleman bulunamazsa -1.

Örnek:

```
//--- CArrayObj:: SearchLast(CObject*) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- diziyi sırala
   array.Sort();
   //--- örneği oluştur
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Örnek oluşturma hatası");
      delete array;
      return;
     }
   //--- örnek özelliklerini ayarla
   //--- . . .
   //--- eleman ara
   if(array.SearchLast(sample)!=-1) printf("Eleman bulundu");
   else                             printf("Eleman bulunamadı");
   //--- diziyi sil
   delete array;
  }
```

Save

Veri dizisini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CArrayObj::Save(int) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayObj *array=new CArrayObj;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- diziye eleman ekle
   //--- . . .
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Save(file_handle))
        {
         //--- dosya kaydetme hatası
         printf("Dosya kaydedilemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   delete array;
  }
```

Load

Veri dizisini bir dosyadan yükler

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Not

Dizi dosyadan okunurken herbir elemanın oluşturulması için CArrayObj::CreateElement(int) çağrısı yapılır.

Örnek:

```
//--- CArrayObj::Load(int) için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   int         file_handle;
   CArrayObj *array=new CArrayObj;
   //---
   if(array!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!array.Load(file_handle))
        {
         //--- dosya yükleme hatası
         printf("Dosya yüklenemedi: Hata %d!",GetLastError());
         delete array;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
```

```
   //--- dizi elemanlarını kullan
   //--- . . .
   //--- diziyi sil
   delete array;
  }
```

Type

Dizinin tip tanımlayıcısını alır.

```
virtual int  Type() const
```

Dönüş Değeri

Dizinin tip tanımlayıcısı (CArrayObj için 7778).

Örnek:

```
//--- CArrayObj::Type() için bir örnek
#include <Arrays\ArrayObj.mqh>
//---
void OnStart()
  {
   CArrayObj *array=new CArrayObj;
   //---
   if(array==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dizi tipini al
   int type=array.Type();
   //--- diziyi sil
   delete array;
  }
```

CList

CList sınıfı, CObject ve onun soyundan gelen sınıf örnekleri için bir dinamik liste sınıfıdır.

Açıklama

CList ,sınıfı CObject ve onun soyundan gelen sınıf örneklerinin bir listesi ile çalışabilmeyi sağlar. Sınıf içerisinde; sıralama, sıralanmış listede arama, eleman silme/ekleme gibi işlevler yer almaktadır. Ayrıca dosya işlemleri için gereken yöntemler de sınıf içerisinde yer almaktadır.

CList sınıfı ile çalışmak bir takım incelikler içerir. Bu sınıf dinamik belleği denetlemek için mekanizmalara sahiptir, bu nedenle liste elemanlarıyla çalışılırken dikkatli olunması gerekir.

Bellek yönetim mekanizmasının incelikleri daha önce CArrayObj konusunda anlatılanlarla benzerdir.

Bildirim

```
class CList : public CObject
```

Başlık

```
#include <Arrays\List.mqh>
```

Kalıtım hiyerarşisi

CObject
  CList

Sınıf Yöntemleri

Özellikler	
FreeMode	Liste elemanlarının silinmesi sırasında kullanılacak bellek yönetim bayrağını alır
FreeMode	Liste elemanlarının silinmesi sırasında kullanılacak bellek yönetim bayrağını ayarlar
Total	Listedeki elemanların toplam sayısını alır
IsSorted	Listenin sıralama bayrağını alır
SortMode	Sıralama türünü alır
Oluşturma yöntemleri	
CreateElement	Yeni bir liste elemanı oluşturur
Ekleme yöntemleri	
Add	Listenin sonuna bir eleman ekler
Insert	Belirtilen liste konumuna eleman ekler
Silme yöntemleri	

Özellikler	
DetachCurrent	Listenin geçerli konumundaki elemanı fiziksel olarak silmeden diziden kaldırır
DeleteCurrent	Listenin mevcut konumundaki elemanı siler
Delete	Listenin belirtilen konumundaki elemanı siler
Clear	Tüm liste elemanlarını kaldırır
Konumlandırma	
IndexOf	Liste elemanının indisini alır
GetNodeAtIndex	Belirtilen konumdaki liste elemanını alır
GetFirstNode	Listedeki ilk elemanı alır
GetPrevNode	Listedeki bir önceki elemanı alır
GetCurrentNode	Mevcut liste elemanını alır
GetNextNode	Listedeki bir sonraki elemanı alır
GetLastNode	Son elemanı alır
Sıralama yöntemleri	
Sort	Listeyi sıralar
MoveToIndex	Mevcut elemanı listedeki belli bir konuma taşır
Exchange	Liste elemanlarının yerlerini karşılıklı değiştirir
Karşılaştırma yöntemleri	
CompareList	Listeyi başka bir liste ile karşılaştırır
Arama yöntemleri	
Search	Sıralı liste içinde, belirtilen kritere eşit olan bir eleman arar
Girdi/çıktı	
virtual Save	Veri listesini dosyaya kaydeder
virtual Load	Veri listesini dosyadan yükler
virtual Type	Listenin tip tanımlayıcısını alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

FreeMode

Liste elemanlarının silinmesi sırasında kullanılacak bellek yönetim bayrağını alır

```
bool  FreeMode() const
```

Dönüş Değeri

Bellek yönetimi bayrağı.

Örnek:

```
//--- CList::FreeMode() için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- serbest durum bayrağını al
   bool list_free_mode=list.FreeMode();
   //--- listeyi sil
   delete list;
  }
```

FreeMode

Liste elemanlarının silinmesi sırasında kullanılacak bellek yönetim bayrağını ayarlar.

```
void  FreeMode(
   bool  mode      // yeni değer
   )
```

Parametreler

mode

[in] Bellek yönetim bayrağı için yeni değer.

Not

Bellek yönetimi bayrağının ayarlanması CList kullanımında önemli bir noktadır. Liste elemanları dinamik nesne işaretçileri olduğundan, listeden kaldırıldıklarında akıbetlerinin ne olacağını belirlemek oldukça önemlidir. Bayrak ayarlanmışsa, elemanın listeden kaldırılmasının ardından 'delete' operatörü ile eleman otomatik olarak silinecektir. Bayrak ayarlanmamışsa, silinen elemanın işaretçisinin hala program içerisinde bir yerlerde olduğu varsayılır ve daha sonra programdan kaldırılacaktır.

Kullanıcının bellek yönetim bayrağını sıfırlaması durumunda, program tamamlanmadan listenin kaldırılması gereklidir, aksi takdirde, 'new' operatörü kullanıldığında bellek boşaltılmamış elemanlarla dolacaktır. Büyük miktarlardaki veriler terminalin çökmesine hatta bozulmasına neden olabilir.

Kullanıcının bellek yönetim bayrağını sıfırlamaması durumunda ise başka bir "tehlike" doğacaktır. İşaretçilerin kullanılması durumunda liste elemanları yerel değişkenler içinde depolanacaktır. Bu durumda listenin kaldırılması kritik hataya yol açacak ve programı çökertecektir. Varsayılan olarak, bellek yönetim bayrağı ayarlıdır (yönetim mekanizması açık), yani belleğin boşaltılmasından liste sınıfı sorumludur.

Örnek:

```
//--- CList::FreeMode(bool) için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- serbest durum bayrağını sıfırla
   list.FreeMode(false);
   //--- listeyi kullan
   //--- . . .
   //--- listeyi sil
   delete list;
```

```
}
```

Total

Listedeki elemanların toplam sayısını alır.

```
int  Total() const
```

Dönüş Değeri

Listedeki elemanların toplam sayısı.

Örnek:

```
//--- CList::Total() için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- toplamı kontrol et
   int total=list.Total();
   //--- listeyi kullan
   //--- ...
   //--- listeyi sil
   delete list;
  }
```

IsSorted

Listenin sıralama bayrağını alır.

```
bool  IsSorted(
   int  mode=0      // Sıralama kipi
   ) const
```

Parametreler

mode=0

[in] Sınanmış sıralama türü

Dönüş Değeri

Listenin sıralama bayrağı. Liste belirtilen şekilde sıralanmışsa 'true', Aksi durumda 'false'.

Not

Listelerin sıralama bayrakları doğrudan değiştirilemez. Sıralanma bayrağı Sort () yöntemi ile ayarlanır ve ekleme yöntemlerini sıfırlar.

Örnek:

```
//--- CList::IsSorted() için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- sıralamayı kontrol et
   if(list.IsSorted(0))
     {
      //--- Sıralı liste yöntemlerini kullan
      //--- ...
     }
   //--- listeyi sil
   delete list;
  }
```

SortMode

Sıralama türünü alır.

```
int  SortMode() const
```

Dönüş Değeri

Başarılı durumda sıralama türüne, liste sıralanmamışsa -1 değerine dönüş yapar.

Örnek:

```
//--- CList::SortMode() için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- sıralama kipini kontrol et
   int sort_mode=list.SortMode();
   //--- listeyi kullan
   //--- ...
   //--- listeyi sil
   delete list;
  }
```

CreateElement

Yeni bir liste elemanı oluşturur.

```
CObject*  CreateElement()
```

Dönüş Değeri

Oluşturulan elemanın işaretçisi. Eleman kaldırılamamışsa NULL dönüşü yapar.

Not

CList sınıfının yöntemi CreateElement () her zaman NULL dönüşü yapar ve herhangi bir işlem gerçekleştirmez. Gerekli olduğu takdirde bir türetik sınıfın CreateElement () yöntemi kullanılmalıdır.

Örnek:

```
//--- CList::CreateElement(int) için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    size=100;
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- fill list
   for(int i=0;i<size;i++)
     {
      CObject *object=list.CreateElement();
      if(object==NULL)
        {
         printf("Eleman oluşturma hatası");
         delete list;
         return;
        }
      list.Add(object);
     }
   //--- listeyi kullan
   //--- . . .
   //--- listeyi sil
   delete list;
  }
```

Add

Listenin sonuna bir eleman ekler.

```
int  Add(
   CObject*  element      // eklenecek eleman
   )
```

Parametreler

element

[in] Listeye eklenecek elemanın değeri.

Dönüş Değeri

Başarı durumunda eklenen elemanın değerine, aksi durumda -1 değerine dönüş yapar.

Not

Geçersiz bir işaretçinin (NULL gibi) parametre olarak geçirilmesi durumunda eleman listeye eklenmeyecektir.

Örnek:

```
//--- CList::Add(Cobject*) için bir örenek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- 100 eleman ekle
   for(int i=0;i<100;i++)
     {
      if(list.Add(new CObject)==-1)
        {
         printf("Eleman ekleme hatası");
         delete list;
         return;
        }
     }
   //--- listeyi kullan
   //--- . . .
   //--- listeyi sil
   delete list;
  }
```

Insert

Belirtilen liste konumuna eleman ekler.

```
int  Insert(
   CObject*  element,      // eklenecek eleman
   int       pos           // konum
   )
```

Parametreler

element

[in] Listeye eklenecek elemanın değeri

pos

[in] Elemanın ekleneceği liste konumu

Dönüş Değeri

Başarı durumunda eklenen elemanın indisine, aksi durumda -1 değerine dönüş yapar.

Not

Geçersiz bir işaretçinin (NULL gibi) parametre olarak geçirilmesi durumunda eleman listeye eklenmeyecektir.

Örnek:

```
//--- CList::Insert(CObject*,int) için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- 100 eleman ekle
   for(int i=0;i<100;i++)
     {
      if(list.Insert(new CObject,0)==-1)
        {
         printf("Eleman ekleme hatası");
         delete list;
         return;
        }
     }
   //--- listeyi kullan
   //--- . . .
```

```
   //--- listeyi sil
   delete list;
  }
```

DetachCurrent

Listenin mevcut konumundaki elemanı fiziksel olarak silmeden diziden kaldırır.

```
CObject*  DetachCurrent()
```

Dönüş Değeri

Kaldırılan elemanın işaretçisi. Eleman kaldırılamamışsa NULL dönüşü yapar.

Not

Eleman listeden ayrıldığında, bellek yönetim bayrağının herhangi bir durumu için bellekten kaldırılmaz. Ayrılan elemanın işaretçisi kullanımdan sonra bellekten silinmelidir.

Örnek:

```
//--- CList::DetachCurrent() için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- liste elemanları ekle
   //--- . . .
   CObject *object=list.DetachCurrent();
   if(object==NULL)
     {
      printf("Ayırma hatası");
      delete list;
      return;
     }
   //--- elemanı kullan
   //--- . . .
   //--- elemanı sil
   delete object;
   //--- listeyi sil
   delete list;
  }
```

DeleteCurrent

Listenin mevcut konumundaki elemanı siler.

```
bool  DeleteCurrent()
```

Dönüş Değeri

Başarılı ise 'true', eleman silinemezse 'false'.

Not

Bellek yönetimi etkinse silinen eleman bellekten de boşaltılır.

Örnek:

```
//--- CList::DeleteCurrent() için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- liste elemanları ekle
   //--- . . .
   if(!list.DeleteCurrent())
     {
      printf("Silme hatası");
      delete list;
      return;
     }
   //--- listeyi sil
   delete list;
  }
```

Delete

Listenin mevcut konumundaki elemanı siler.

```
bool  Delete(
   int  pos      // konum
   )
```

Parametreler

pos

[in] Silinecek elemanın konumu.

Dönüş Değeri

Başarılı ise 'true', eleman silinemezse 'false'.

Not

Bellek yönetimi etkinse silinen eleman bellekten de boşaltılır.

Örnek:

```
//--- CList::Delete(int) için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- liste elemanları ekle
   //--- . . .
   if(!list.Delete(0))
     {
      printf("Silme hatası");
      delete list;
      return;
     }
   //--- listeyi sil
   delete list;
  }
```

Clear

Listenin tüm elemanlarını siler.

```
void  Clear()
```

Not

Bellek yönetimi mekanizması açıksa, silinene elemanlar bellekten boşaltılır.

Örnek:

```
//--- CList::Clear() için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- liste elemanları ekle
   //--- . . .
   //--- listeyi temizle
   list.Clear();
   //--- listeyi sil
   delete list;
  }
```

IndexOf

Liste elemanının indisini alır.

```
int  IndexOf(
   CObject*  element      // elemanın işaretçisi
   )
```

Parametreler

element

[in] Liste elemanının işaretçisi.

Dönüş Değeri

Başarılı ise liste elemanının indisine, aksi durumda -1 değerine dönüş yapar.

Örnek:

```
//--- CList::IndexOf(CObject*) için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   CObject *object=new CObject;
   if(object==NULL)
     {
      printf("Eleman oluşturma hatası");
      delete list;
      return;
     }
   if(list.Add(object))
     {
      int pos=list.IndexOf(object);
     }
   //--- listeyi sil
   delete list;
  }
```

GetNodeAtIndex

Belirtilen konumdaki liste elemanını alır.

```
CObject*  GetNodeAtIndex(
   int  pos      // konum
   )
```

Parametreler

pos

[in] Liste elemanının indisi.

Dönüş değeri

Başarılı ise elemanın işaretçisine, aksi durumda NULL değerine dönüş yapar.

Örnek:

```
//--- CList::GetNodeAtIndex(int) için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- liste elemanları ekle
   //--- . . .
   CObject *object=list.GetNodeAtIndex(10);
   if(object==NULL)
     {
      printf("Eleman alınamadı");
      delete list;
      return;
     }
   //--- elemanı kullan
   //--- . . .
   //--- elemanı silme
   //--- listeyi sil
   delete list;
  }
```

GetFirstNode

Listedeki ilk elemanı alır.

```
CObject*  GetFirstNode()
```

Dönüş Değeri

Başarılı ise ilk elemanın işaretçisine, aksi durumda NULL değerine dönüş yapar.

Örnek:

```
//--- CList::GetFirstNode() için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- liste elemanları ekle
   //--- . . .
   CObject *object=list.GetFirstNode();
   if(object==NULL)
     {
      printf("Eleman alınamadı");
      delete list;
      return;
     }
   //--- elemanı kullan
   //--- . . .
   //--- elemanı silme
   //--- listeyi sil
   delete list;
  }
```

GetPrevNode

Listedeki bir önceki elemanı alır.

```
CObject*  GetPrevNode()
```

Dönüş Değeri

Başarılı ise bir önceki liste elemanının işaretçisine, aksi durumda NULL değerine dönüş yapar.

Örnek:

```
//--- CList::GetPrevNode() için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- liste elemanları ekle
   //--- . . .
   CObject *object=list.GetPrevNode();
   if(object==NULL)
     {
      printf("Eleman alınamadı");
      delete list;
      return;
     }
   //--- elemanı kullan
   //--- . . .
   //--- elemanı silme
   //--- listeyi sil
   delete list;
  }
```

GetCurrentNode

Mevcut liste elemanını alır.

```
CObject*  GetCurrentNode()
```

Dönüş Değeri

Başarılı ise mevcut elemanın işaretçisine, aksi durumda NULL değerine dönüş yapar.

Örnek:

```
//--- CList::GetCurrentNode() için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- liste elemanları ekle
   //--- . . .
   CObject *object=list.GetCurrentNode();
   if(object==NULL)
     {
      printf("Eleman alınamadı");
      delete list;
      return;
     }
   //--- elemanı kullan
   //--- . . .
   //--- elemanı silme
   //--- listeyi sil
   delete list;
  }
```

GetNextNode

Listedeki bir sonraki elemanı alır.

```
CObject*  GetNextNode()
```

Dönüş Değeri

Başarılı ise bir sonraki elemanın işaretçisine, aksi durumda NULL değerine dönüş yapar.

Örnek:

```
//--- CList::GetNextNode() için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- liste elemanları ekle
   //--- . . .
   CObject *object=list.GetNextNode();
   if(object==NULL)
     {
      printf("Eleman alınamadı");
      delete list;
      return;
     }
   //--- elemanı kullan
   //--- . . .
   //--- elemanı silme
   //--- listeyi sil
   delete list;
  }
```

GetLastNode

Listedeki son elemanı alır.

```
CObject*  GetLastNode()
```

Dönüş Değeri

Başarılı ise son elemanın işaretçisine, aksi durumda NULL değerine dönüş yapar.

Örnek:

```
//--- CList::GetLastNode() için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- liste elemanları ekle
   //--- . . .
   CObject *object=list.GetLastNode();
   if(object==NULL)
     {
      printf("Eleman alınamadı");
      delete list;
      return;
     }
   //--- elemanı kullan
   //--- . . .
   //--- elemanı silme
   //--- listeyi sil
   delete list;
  }
```

Sort

Listeyi sıralar.

```
void  Sort(
   int  mode      // sıralama kipi
   )
```

Parametreler

mode

[in] Sıralama yöntemi.

Dönüş Değeri

Yok.

Not

Liste sıralamaları her zaman artan türdedir.

Örnek:

```
//--- CList::Sort(int) için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- 0 modu ile sıralama
   list.Sort(0);
   //--- listeyi kullan
   //--- ...
   //--- listeyi sil
   delete list;
  }
```

MoveToIndex

Mevcut elemanı listedeki belli bir konuma taşır.

```
bool  MoveToIndex(
   int  pos      // konum
   )
```

Parametreler

pos

[in] Elemanın taşınacağı yeni liste konumu.

Dönüş Değeri

Başarılı ise 'true', eleman kaydırılamadıysa 'false'.

Örnek:

```
//--- CList::MoveToIndex(int) için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- mevcut elemanı başa taşı
   list.MoveToIndex(0);
   //--- listeyi kullan
   //--- . . .
   //--- listeyi sil
   delete list;
  }
```

Exchange

Liste elemanlarının yerlerini karşılıklı değiştirir.

```
bool  Exchange(
   CObject*  node1,     // liste elemanı
   CObject*  node2      // liste elemanı
   )
```

Parametreler

node1

[in] Liste elemanı

node2

[in] Liste elemanı

Dönüş Değeri

Başarılı ise 'true', eleman değiştirilemezse 'false'.

Örnek:

```
//--- CList::Exchange(CObject*,CObject*) için bir örenek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- değişim
   list.Exchange(list.GetFirstNode(),list.GetLastNode());
   //--- listeyi kullan
   //--- . . .
   //--- listeyi sil
   delete list;
  }
```

CompareList

Listeyi başka bir liste ile karşılaştırır.

```
bool  CompareList(
   CList*  list      // karşılaştırılacak liste
   )
```

Parametreler

list

[in] Bir CList sınıf örneğinin işaretçisi (karşılaştırma için kaynak elemanlar).

Dönüş Değeri

listeler eşitse 'true', değilse 'false'.

Örnek:

```
//--- CList::CompareList(const CList*) için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- kaynak listeyi oluştur
   CList *src=new CList;
   if(src==NULL)
     {
      printf("Nesne oluşturma hatası");
      delete list;
      return;
     }
   //--- kaynak listeyi doldur
   //--- . . .
   //--- başka bir liste ile karşılaştır
   bool result=list.CompareList(src);
   //--- listeleri sil
   delete src;
   delete list;
  }
```

Search

Sıralı liste içinde, belirtilen kritere eşit olan bir eleman arar.

```
CObject*  Search(
   CObject*  element      // örnek
   )
```

Parametreler

element

[in] Dizi içinde aranacak elemanın örneği.

Dönüş Değeri

Başarılı ise bulunan elemanın işaretçisine, aksi durumda NULL değerine dönüş yapar.

Örnek:

```
//--- CList::Search(CObject*) için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- iste elemanları ekle
   //--- . . .
   //--- listeyi sırala
   list.Sort(0);
   //--- liste örnek oluştur
   CObject *sample=new CObject;
   if(sample==NULL)
     {
      printf("Örnek oluşturma hatası");
      delete list;
      return;
     }
   //--- örnek özelliklerini ayarla
   //--- . . .
   //--- eleman ara
   if(list.Search(sample)!=NULL) printf("Eleman bulundu");
   else                          printf("Eleman bulunamadı");
   //--- listeyi sil
   delete list;
  }
```

Save

Veri listesini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CList::Save(int) için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CList *list=new CList;
   //---
   if(list!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- iste elemanları ekle
   //--- . . .
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!list.Save(file_handle))
        {
         //--- dosya kaydetme hatası
         printf("Dosya kaydedilemedi: Hata %d!",GetLastError());
         delete list;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- listeyi sil
   delete list;
```

```
}
```

Load

Liste verisini dosyadan yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen (...) fonksiyonu ile açılmış olan dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Not

Liste dosyadan okunurken herbir elemanın oluşturulması için CList:: CreateElement () çağrısı yapılır.

Örnek:

```
//--- CLoad::Load(int) için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   int    file_handle;
   CList *list=new CList;
   //---
   if(list!=NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!list.Load(file_handle))
        {
         //--- dosya yükleme hatası
         printf("Dosya yükleme başarısız. Hata %d!",GetLastError());
         delete list;
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- liste elemanlarını kullan
```

```
   //--- . . .
   //--- listeyi sil
   delete list;
  }
```

Type

Listenin tip tanımlayıcısını alır.

```
virtual int  Type()
```

Dönüş Değeri

Tip tanımlayıcısı (CList için 7779).

Örnek:

```
//--- CList::Type() için bir örnek
#include <Arrays\List.mqh>
//---
void OnStart()
  {
   CList *list=new CList;
   //---
   if(list==NULL)
     {
      printf("Nesne oluşturma hatası");
      return;
     }
   //--- liste tipini al
   int type=list.Type();
   //--- listeyi sil
   delete list;
  }
```

CTreeNode

CTreeNode sınıfı, ikili CTree ağacının düğüm sınıfıdır.

Açıklama

CTreeNode sınıfı, CTree ikili ağacının düğümleriyle çalışmaya olanak sağlar. Sınıf içinde, ağaç üzerinde konumlandırma yapmak için seçenekler bulunur. Ayrıca dosya işlemleri için gereken yöntemler de sınıf içerisinde yer almaktadır.

Bildirim

```
class CTreeNode : public CObject
```

Başlık

```
#include <Arrays\TreeNode.mqh>
```

Kalıtım hiyerarşisi

CObject

CTreeNode

İlk nesil

CTree

Sınıf Yöntemleri

Özellikler	
Owner	Düğüm sahibinin işaretçisini alır/ayarlar
Left	Sol düğümün işaretçisini alır/ayarlar
Right	Sağ düğümün işaretçisini alır/ayarlar
Balance	Düğüm dengesini alır
BalanceL	Düğümün sol-alt dalının dengesini alır
BalanceR	Düğümün sağ-alt dalının dengesini alır
Yeni eleman oluşturulması	
CreateSample	Yeni bir düğüm örneği oluşturur
Karşılaştırma	
RefreshBalance	Düğüm dengesini yeniden hesaplar
Arama	
GetNext	Sonraki düğümün işaretçisini alır
Giriş/Çıkış	

Özellikler	
SaveNode	Düğüm verilerini bir dosyaya kaydeder
LoadNode	Düğüm verilerini bir dosyadan yükler
virtual Type	Düğümün tip tanımlayıcısını alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Compare

Ağaçlara dair pratik uygulamalar CTreeNode sınıfının soyundan gelir.

CTreeNode sınıfının soyundan gelen her örnek şu ön-tanımlı yöntemleri içermelidir: CreateSample (CTreeNode soyundan gelen sınıfın bir örneğini oluşturur), Compare (CTreeNode soyundan gelen sınıfın anahtar bölgelerindeki değerleri karşılaştırır), Type (düğümü tanımlamak gerektiğinde kullanılır), SaveNode ve LoadNode (dosya işlemleri gerektiğinde kullanılır).

CTree soyundan gelen bir sınıf örneğini ele alalım.

```
//+------------------------------------------------------------------+
//|                                                   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>
//+------------------------------------------------------------------+
//| CTreeNode'dan türeyen bir sınıf tanımla.                         |
//+------------------------------------------------------------------+
//| CMyTreeNode sınıfı.                                              |
//| Amaç: bir ikili ağaç elemanının sınıfı.                          |
//|              CTreeNode sınıfının bir türevi.                     |
//+------------------------------------------------------------------+
class CMyTreeNode : public CTreeNode
  {
protected:
   //--- kullanıcı verisi
   long              m_long;           // long tipi için anahtar alan
   double            m_double;         // double tipli, kullanıcı-tanımlı değişken
   string            m_string;         // string tipli, kullanıcı-tanımlı değişken
   datetime          m_datetime;       // datetime tipli, kullanıcı-tanımlı değişken

public:
                     CMyTreeNode();
   //--- kullanıcı verisine ulaşma yöntemleri
   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;  }
   //--- dosyalarla çalışmak için yöntemler
   virtual bool      Save(int file_handle);
   virtual bool      Load(int file_handle);
protected:
   virtual int       Compare(const CObject *node,int mode);
   //--- sınıf örnekleri oluşturmak için yöntemler
   virtual CTreeNode* CreateSample();
  };
//+------------------------------------------------------------------+
//| CMyTreeNode sınıf yapıcısı.                                      |
//| INPUT:  yok.                                                     |
//| OUTPUT: yok.                                                     |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
void CMyTreeNode::CMyTreeNode()
  {
//--- kullanıcı verisinin başlatılması
   m_long        =0;
   m_double      =0.0;
   m_string      ="";
   m_datetime    =0;
  }
//+------------------------------------------------------------------+
//| Belirtilen algoritmayla, ağaç düğümlerinin karşılaştırılması.    |
//| INPUT:  node - karşılaştırılacak dizi elemanı,                   |
//|         mode - kıyaslama algoritmasının tanımlayıcısı.           |
//| OUTPUT: karşılaştırma sonucu (>0,0,<0).                          |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
int CMyTreeNode::Compare(const CObject *node,int mode)
  {
//--- başka kıyaslama algoritması olmadığından 'mode' parametresi göz-ardı edildi
   int res=0;
//--- açık tip dönüşümü
   CMyTreeNode *n=node;
   res=(int)(m_long-n.m_long);
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| Yeni bir sınıf örneği oluşturma.                                 |
//| INPUT:  yok.                                                     |
//| OUTPUT: yeni CMyTreeNode sınıf örneğinin işaretçisi.             |
```

```
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
CTreeNode* CMyTreeNode::CreateSample()
  {
   CMyTreeNode *result=new CMyTreeNode;
//---
   return(result);
  }
//+------------------------------------------------------------------+
//| Düğüm verisini dosyaya yaz.                                      |
//| INPUT:  file_handle - açılmış olan dosyanın tanıtıcı değeri.     |
//| OUTPUT: doğruysa true, değilse false.                            |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CMyTreeNode::Save(int file_handle)
  {
   uint i=0,len;
//--- denetle
   if(file_handle<0) return(false);
//--- writing user data
//--- long tipli kullanıcı tanımlı değişkeni yaz
   if(FileWriteLong(file_handle,m_long)!=sizeof(long))           return(false);
//--- double tipli kullanıcı tanımlı değişkeni yaz
   if(FileWriteDouble(file_handle,m_double)!=sizeof(double))     return(false);
//--- string tipli kullanıcı tanımlı değişkeni yaz
   len=StringLen(m_string);
//--- dizgi uzunluğunu yaz
   if(FileWriteInteger(file_handle,len,INT_VALUE)!=INT_VALUE)    return(false);
//--- dizgiyi yaz
   if(len!=0 && FileWriteString(file_handle,m_string,len)!=len) return(false);
//--- datetime tipli kullanıcı tanımlı değişkeni yaz
   if(FileWriteLong(file_handle,m_datetime)!=sizeof(long))       return(false);
//---
   return(true);
  }
//+------------------------------------------------------------------+
//| Düğüm verisini dosyadan oku.                                     |
//| INPUT:  file_handle - açılmış olan dosyanın tanıtıcı değeri.     |
//| OUTPUT: doğruysa true, değilse false.                            |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CMyTreeNode::Load(int file_handle)
  {
   uint i=0,len;
//--- denetle
   if(file_handle<0) return(false);
//--- oku
   if(FileIsEnding(file_handle)) return(false);
//--- char tipli kullanıcı tanımlı değişkenin okunması
```

```
//--- long tipli kullanıcı tanımlı değişkenin okunması
   m_long=FileReadLong(file_handle);
//--- double tipli kullanıcı tanımlı değişkenin okunması
   m_double=FileReadDouble(file_handle);
//--- string tipli kullanıcı tanımlı değişkenin okunması
//--- dizgi uzunluğunu oku
   len=FileReadInteger(file_handle,INT_VALUE);
//--- dizgiyi oku
   if(len!=0) m_string=FileReadString(file_handle,len);
   else       m_string="";
//--- datetime tipli kullanıcı tanımlı değişkenin okunması
   m_datetime=FileReadLong(file_handle);
//---
   return(true);
  }
```

Owner

Düğüm sahibinin işaretçisini alır.

```
CTreeNode*  Owner()
```

Dönüş Değeri

Düğüm sahibinin işaretçi.

Owner

Düğüm sahibinin işaretçisini ayarlar.

```
void  Owner(
   CTreeNode*  node       // düğüm
   )
```

Parametreler

node

[in] Düğüm sahibinin işaretçisi için yeni değer.

Dönüş Değeri

Yok.

Left

Sol düğümün işaretçisini alır.

```
CTreeNode*  Left()
```

Dönüş Değeri

Sol düğümün işaretçisi.

Left

Sol düğümün işaretçisini ayarlar.

```
void  Left(
   CTreeNode*  node       // düğüm
   )
```

Parametreler

node

[in] Sol düğümün işaretçisi için yeni değer.

Dönüş Değeri

Yok.

Right

Sağ düğümün işaretçisini alır.

```
CTreeNode*  Right()
```

Dönüş Değeri

Sağ düğümün işaretçisi.

Right

Sağ düğümün işaretçisini ayarlar.

```
void  Right(
   CTreeNode*  node       // düğüm
   )
```

Parametreler

node

[in] Sağ düğümün işaretçisi için yeni değer.

Dönüş Değeri

Yok.

Balance

Düğümün denge bilgisini alır.

```
int  Balance() const
```

Dönüş Değeri

Düğüm dengesi.

BalanceL

Düğümün sol-alt dalının dengesini alır.

```
int  BalanceL() const
```

Dönüş Değeri

Düğümün sol-alt dalının dengesi.

BalanceR

Düğümün sağ-alt dalının dengesini alır.

```
int  BalanceR() const
```

Dönüş Değeri

Düğümün sağ-alt dalının dengesi.

CreateSample

Yeni bir düğüm örneği oluşturur.

```
virtual CTreeNode*  CreateSample()
```

Dönüş Değeri

Yeni düğüm örneğinin işaretçisi veya NULL.

RefreshBalance

Düğüm dengesini yeniden hesaplar.

```
int  RefreshBalance()
```

Dönüş Değeri

Düğüm dengesi.

GetNext

Sonraki düğümün işaretçisini alır.

```
CTreeNode*  GetNext(
   CTreeNode*  node       // düğüm
   )
```

Parametreler

node

[in] Arama başlangıç düğümü.

Dönüş Değeri

Sonraki düğümün işaretçisi.

SaveNode

Düğüm verisini dosyaya yazar.

```
bool  SaveNode(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] Yazma için açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

LoadNode

Düğüm verisini dosyadan okur.

```
bool  LoadNode(
   int          file_handle,     // dosya tanıtıcı değeri
   CTreeNode*   main             // düğüm
   )
```

Parametreler

file_handle

[in] Okuma için açılmış olan ikili dosyanın tanıtıcı değeri.

main

[in] Veri düğümü.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Düğümün tip tanımlayıcısını alır.

```
virtual int  Type() const
```

Dönüş Değeri

Düğümün tip tanımlayıcısı.

CTree

CTree; CTreeNode ve onun soyundan gelen sınıf örneklerinin ikili ağaçları için tasarlanmış bir sınıftır.

Açıklama

CTree sınıfı, CTreeNode ve onun soyundan gelen sınıf örneklerinin ikili ağaçları ile çalışmaya olanak sağlar. Ağaç elemanlarının eklenip silinmesi ve arama gibi seçenekler sınıf içerisinde uygulanmıştır. Ayrıca dosya işlemleri için gereken yöntemler de sınıf içerisinde yer almaktadır.

CList ve CArrayObj sınıflarından farklı olarak, bellek yönetim mekanizması CTree içerisinde uygulanmamıştır. Tüm ağaç düğümleri bellek temizleme işlemi ile silinir.

Bildirim

```
class CTree : public CTreeNode
```

Başlık

```
#include <Arrays\Tree.mqh>
```

Kalıtım hiyerarşisi

CObject
  CTreeNode
    CTree

Sınıf Yöntemleri

Özellikler	
Root	Ağacın kök elemanını alır
Yeni eleman oluşturulması	
CreateElement	Yeni bir düğüm örneği oluşturur
Doldurma	
Insert	Ağaca yeni bir düğüm ekler
Silme	
Detach	Belli bir düğümü ağaçtan ayırır
Delete	Belirtilen bir düğümü ağaçtan siler
Clear	Ağacın tüm düğümlerini temizler
Arama	
Find	Ağaç içerisinde, örneğe göre bir düğüm arar
Girdi/çıktı	

Özellikler	
virtual Save	Tüm ağaç verisini dosyaya kaydeder
virtual Load	Ağaç verisini dosyadan yükler
virtual Type	Ağacın tip tanımlayıcısını alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CTreeNode

Parent, Parent, Left, Left, Right, Right, Balance, BalanceL, BalanceR, RefreshBalance, GetNext, SaveNode, LoadNode

Ağaçlarla ilgi pratik uygulamalar CTreeNode sınıfının - ve dolayısıyla CTree - sınıfının soyundan gelir.

CTree'nin soyundan gelen sınıf, CreateElement ön-tanımlı yöntemini içermelidir. Bu yöntem CTreeNode türetik sınıfının örneklerini oluşturur.

CTree sınıfından türetilen bir örneğe bakalım:

```
//+------------------------------------------------------------------+
//|                                                      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;
//+------------------------------------------------------------------+
//| CTree'den türetilen CMyTree sınıfını tanımla.                    |
//+------------------------------------------------------------------+
//| CMyTree sınıfı                                                   |
//| Amaç: İkili arama ağacının inşaası ve konumlandırılması.         |
//+------------------------------------------------------------------+
class CMyTree : public CTree
  {
public:
   //--- Ağaç üzerinde kullanıcı tanımlı arama yöntemleri
   CMyTreeNode*         FindByLong(long find_long);
   //--- Ağaç elemanı oluşturma yöntemi
   virtual CTreeNode *CreateElement();
  };
//---
CMyTree MyTree;
```

```
//+------------------------------------------------------------------+
//| Yeni bir ağaç elemanı oluşturma.                                 |
//| INPUT:  yok.                                                     |
//| OUTPUT: yeni ağaç düğümünün işaretçisi OK veya NULL.             |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
CTreeNode *CMyTree::CreateElement()
  {
   CMyTreeNode *node=new CMyTreeNode;
//---
   return(node);
  }
//+------------------------------------------------------------------+
//| m_long değerini kullanarak listede bir eleman ara.               |
//| INPUT:  find_long - aranan değer.                                |
//| OUTPUT: bulunan liste elemanının işaretçisi veya NULL.           |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
CMyTreeNode* CMyTree::FindByLong(long find_long)
  {
   CMyTreeNode *res=NULL;
   CMyTreeNode *node;
//--- arama parametresini geçirebilmek için bir ağaç düğümü oluştur
   node=new CMyTreeNode;
   if(node==NULL) return(NULL);
   node.SetLong(find_long);
//---
   res=Find(node);
   delete node;
//---
   return(res);
  }
//+------------------------------------------------------------------+
//| script "CMyTree sınıfı için bir sınama"                          |
//+------------------------------------------------------------------+
//---  dizgi başlatma için bir bir dizi
string str_array[11]={"p","oo","iii","uuuu","yyyyy","ttttt","rrrr","eee","ww","q","99
//---
int OnStart() export
  {
   int          i;
   uint         pos;
   int          beg_time,end_time;
   CMyTreeNode *node; //--- CMyTreeNode sınıf örneği için geçici işaretçi
//---
   printf("Sınamayı başlat %s.",__FILE__);
//--- MyTree ağacını MyTreeNode sınıfının örnekleri ile extCountedNodes sayısı kadar
   beg_time=GetTickCount();
   for(i=0;i<extCountedNodes;i++)
```

```
      {
       node=MyTree.CreateElement();
       if(node==NULL)
         {
          //--- acil çıkış
          printf("%s (%4d): oluşturma hatası",__FILE__,__LINE__);
          return(__LINE__);
         }
       NodeSetData(node,i);
       node.SetLong(i);
       MyTree.Insert(node);
      }
   end_time=GetTickCount();
   printf("MyTree doldurma zamanı: %d ms.",end_time-beg_time);
//--- TmpMyTree geçici ağacını oluştur.
   CMyTree TmpMyTree;
//--- Ağaç elemanlarının 50%'sini (tek olanları) ağaçtan ayır
//--- ve onları geçici olarak TmpMyTree ağacına ekle.
   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(" %d elemanın MyTree ağacıdan silinme zamanı %d ms.",extCountedNodes/2,end_t
//--- Ayrılan elemanlara dönüş yap
   node=TmpMyTree.Root();
   while(node!=NULL)
     {
      if(TmpMyTree.Detach(node)) MyTree.Insert(node);
      node=TmpMyTree.Root();
     }
//--- Save(int file_handle) yönteminin işleyişini kontrol et
   int file_handle;
   file_handle=FileOpen("MyTree.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!MyTree.Save(file_handle))
        {
         //--- dosya yazma hatası
         //--- acil çıkış
         printf("%s: Hata %d, %d numaralı satır!",__FILE__,GetLastError(),__LINE__);
         //--- çıkmadan önce dosyayı kapa!!!
         FileClose(file_handle);
         return(__LINE__);
        }
      FileClose(file_handle);
```

```
      }
//--- Load(int file_handle) yönteminin işleyişini kontrol et
   file_handle=FileOpen("MyTree.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!TmpMyTree.Load(file_handle))
        {
         //--- dosya okuma hatası
         //--- acil çıkış
         printf("%s: Hata %d, %d numaralı satır!",__FILE__,GetLastError(),__LINE__);
         //--- çıkmadan önce dosyayı kapa!!!
         FileClose(file_handle);
         return(__LINE__);
        }
      FileClose(file_handle);
     }
//---
   MyTree.Clear();
   TmpMyTree.Clear();
//---
   printf("testi sonlandır %s. OK!",__FILE__);
//---
   return(0);
  }
//+------------------------------------------------------------------+
//| Düğüm içeriklerini günlüğe atma fonksiyonu                       |
//+------------------------------------------------------------------+
void NodeToLog(CMyTreeNode *node)
  {
   printf("   %I64d,%f,'%s','%s'",
               node.GetLong(),node.GetDouble(),
               node.GetString(),TimeToString(node.GetDateTime()));
  }
//+------------------------------------------------------------------+
//| Elemanları rassal değerlerle "doldurma" fonksiyonu               |
//+------------------------------------------------------------------+
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

Ağacın kök düğümünü (elemanını) alır.

```
CTreeNode*  Root() const
```

Dönüş Değeri

Ağacın kök düğümünün işaretçisi.

CreateElement

Yeni bir ağaç düğümü (elemanı) oluşturur.

```
virtual CTreeNode*  CreateElement()
```

Dönüş Değeri

Yeni düğümün işaretçisi veya (başarısızsa) NULL.

Insert

Ağaca yeni bir düğüm ekler.

```
CTreeNode*  Insert(
   CTreeNode*  new_node      // yeni düğüm
   )
```

Parametreler

new_node

[in] Ağaca eklenecek düğümün işaretçisi.

Dönüş Değeri

Yeni düğümün işaretçisi veya (başarısızsa) NULL.

Detach

Belirtilen düğümü ağaçtan ayırır.

```
bool  Detach(
   CTreeNode*  node       // düğüm
   )
```

Parametreler

node

[in] Ayrılacak düğüm.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Ayırma işleminin ardından düğüm işaretçisi bellekten kaldırılmaz. Ağaç dengelenir.

Delete

Ağaçtan belli bir düğümü siler.

```
bool  Delete(
   CTreeNode*  node       // düğüm
   )
```

Parametreler

node

[in] Silinecek düğümün işaretçisi.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Silme işleminin ardından düğümün işaretçisi tahliye edilir (bellek boşaltılır). Ağaç dengelenir.

Clear

Ağacın tüm düğümlerini siler.

```
void  Clear()
```

Dönüş Değeri

Yok.

Not

Silme işleminin ardından düğüm işaretçileri tahliye edilir (bellek boşaltılır).

Find

Ağaç içerisinde, örneğe göre bir düğüm arar.

```
CTreeNode*  Find(
   CTreeNode*  node       // düğüm
   )
```

Parametreler

node

[in] Aranan örneği içeren düğüm.

Dönüş Değeri

Yeni düğümün işaretçisi veya (başarısızsa) NULL.

Save

Ağaç verisini dosyaya yazar.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] Yazma için açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Ağaç verisini dosyadan yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] Okuma için açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Ağacın tip tanımlayıcısını alır.

```
virtual int  Type() const
```

Dönüş Değeri

Ağacın tip tanımlayıcısı.

Jenerik Data Koleksiyonları

Kütüphane, kullanıcıların güçlü bir şekilde yazılan koleksiyonlar oluşturmasına izin veren jenerik koleksiyonları tanımlayan sınıflar ve arayüzler sağlar. Bu koleksiyonlar, jenerik olmayan yazılı koleksiyonlara göre daha fazla kolaylık ve veri taşıma performansı sağlar.

Kitaplık, terminal çalışma dizininin Include\Generic klasöründe bulunur.

Nesneler:

Nesne	Tanım	Tür
IKoleksiyon	Jenerik veri koleksiyonlarını uygulama için arayüz	ARAYÜZ
IEqualityComparable	Karşılaştırılabilen nesneleri uygulama için arayüz	ARAYÜZ
IComparable	"Daha büyük, daha düşük veya eşit" olarak karşılaştırılabilen nesneler için arayüz	ARAYÜZ
IKarşılaştırıcı	T türü iki nesneyi birinin diğerinden "daha büyük, daha düşük veya eşit" olup olmadığını karşılaştıran jenerik bir sınıf uygulamak için arayüz.	ARAYÜZ
IEqualityComparer	Varlık için T türündeki iki nesneyi karşılaştıran jenerik bir sınıf uygulamak için arayüz	ARAYÜZ
IListe	Jenerik veri listelerini uygulama arayüzü	ARAYÜZ
IHarita	Anahtar/değer çifti jenerik koleksiyonları uygulamak için arayüz	ARAYÜZ
ISet	Jenerik veri setleri uygulama için arayüz	ARAYÜZ
CDefaultComparer	Compare global metodlarına dayanan IComparer<T> jenerik arayüzünü uygulayan yardımcı bir sınıf	SINIF
CDefaultEqualityComparer	Equals<T> and GetHashCode global metodlarını kullanarak IEqualityComparer<T> jenerik arayüzünü uygulayan yardımcı bir sınıf	SINIF
CArrayList	IList<T> arayüzünü uygulayan jenerik bir sınıf	SINIF
CKeyValuePair	Sınıf, anahtar/değer çiftini uygular.	SINIF
CHashMap	IMap<TKey, TValue> arayüzünü uygulayan jenerik bir sınıf	SINIF

Nesne	Tanım	Tür
CHashSet	ISet<T> arayüzünü uygulayan jenerik bir sınıf	SINIF
CLinkedListNode	CLinkedListNode<T> sınıfını uygulayan bir yardımcı sınıf	SINIF
CLinkedList	ICollection<T> arayüzünü uygulayan bir jenerik sınıf	SINIF
CQueue	ICollection<T> arayüzünü uygulayan bir jenerik sınıf	SINIF
CRedBlackTreeNode	CRedBlackTree<T> sınıfını uygulamasında kullanılan bir yardımcı sınıf	SINIF
CRedBlackTree	ICollection<T> arayüzünü uygulayan bir jenerik sınıf	SINIF
CSortedMap	IMap<TKey, TValue> arayüzünü uygulayan jenerik bir sınıf	SINIF
CSortedSet	ISet<T> arayüzünü uygulayan jenerik bir sınıf	SINIF
CStack	ICollection<T> arayüzünü uygulayan bir jenerik sınıf	SINIF

Global Metodlar:

Metot	Tanım
ArrayBinarySearch	Öğeleri karşılaştırmak için IComparable <T> arayüzünü kullanarak artan sıralı bir boyutlu bir dizide belirtilen değeri arar.
ArrayIndexOf	Tek boyutlu bir dizideki bir değerin ilk ortaya çıkışını arar
ArrayLastIndexOf	Tek boyutlu bir dizideki bir değerin son ortaya çıkışını arar
ArrayReverse	Tek boyutlu bir dizideki öğelerin sırasını değiştirir
Compare	Biri diğerinden daha büyük mü, daha küçük mü veya eşit mi şeklinde iki değeri karşılaştırır
Equals	Varlık için iki değeri karşılaştırır.
GetHashCode	Hash kod değerini hesaplar

ICollection<T>

ICollection<T>, jenerik veri koleksiyonları uygulamak için bir arayüzdür.

Tanım

ICollection <T> arabirimi, öğeleri saymak, bir koleksiyonu temizlemek, öğeler eklemek veya silmek için yöntemler ve diğerlerini de içeren yöntemler de dahil olmak üzere koleksiyonlarla çalışmak için temel yöntemlerini belirler.

Deklarasyon

```
template<typename T>
interface ICollection
```

Başlık

```
#include <Generic\Interfaces\ICollection.mqh>
```

Inheritance Hiyerarşisi

IKoleksiyon

Direct descendants

CLinkedList, CQueue, CRedBlackTree, CStack, IList, IMap, ISet

Sınıf Metodları

Metot	Tanım
Add	Bir koleksiyona bir öğeyi ekler
Count	Bir koleksiyondali öğelerin sayısını döndürür
Contains	Bir koleksiyonun belirli bir değerdeki bir öğeyi içerip içermediğini belirler
CopyTo	Bir koleksiyonun tüm öğelerini, belirtilen indekste başlayan belirtilen diziye kopyalar
Clear	Koleksiyondaki tüm öğeleri kaldırır
Remove	Bir koleksiyondan belirli bir öğenin ilk varlığını kaldırır

Add

Bir koleksiyona bir öğeyi ekler.

```
bool Add(
   T  value     // öğenin değeri
   );
```

Parametreler

value

[in] Eklenecek öğenin değeri

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Count

Bir koleksiyondaki öğelerin sayısını döndürür.

```
int Count();
```

Dönen Değer

Öğelerin sayısını döndürür.

Contains

Bir koleksiyonun belirli bir değerdeki bir öğeyi içerip içermediğini belirler.

```
bool Contains(
   T  item     // arama değeri
   );
```

Parametreler

item

[in] Aranan değer.

Dönen Değer

Eğer belirli bir değerdeki bir öğe bir koleksiyonda bulunuyorsa true, aksi halde false döndürür.

CopyTo

Bir koleksiyonun tüm öğelerini, belirtilen indekste başlayan belirtilen diziye kopyalar.

```
int CopyTo(
   T&          dst_array[],     // yazmak için bir dizi
   const int   dst_start=0      // yazmak için başlangıç indeksi
   );
```

Parametreler

&dst_array[]

[out] Koleksiyonun öğelerinin yazılacağı bir dizi.

dst_start=0

[in] Kopyalamanın başladığı dizideki bir indeks.

Dönen Değer

Kopyalanan öğelerin sayısını döndürür.

Clear

Bir koleksiyonun tüm öğelerini kaldırır.

```
void Clear();
```

Remove

Bir koleksiyondan belirli öğenin ilk varlığını kaldırır.

```
bool Remove(
   T  item     // öğe değeri
   );
```

Parametreler

item

[in] Silinecek öğenin değeri.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

IEqualityComparable<T>

IEqualityComparable<T>, karşılaştırılabilen nesneleri uygulamak için kullanılan bir arabirimdir.

Tanım

IEqualityComparable <T> arabirimi, geçerli nesnenin hash kodunu almak ve aynı türdeki başka bir nesneye eşit olup olmadığını kontrol etmek için yöntemleri tanımlar.

Deklarasyon

```
template<typename T>
interface IEqualityComparable
```

Başlık

```
#include <Generic\Interfaces\IEqualityComparable.mqh>
```

Inheritance Hiyerarşisi

IEqualityComparable

Direct descendants

IComparable

Sınıf Metodları

Metot	Tanım
Equals	Belirli değerdeki geçerli nesneyi karşılaştırır
HashCode	Geçerli nesne için hash kod değerini hesaplar

Equals

Belirli değerdeki geçerli öğeyi karşılaştırır.

```
bool Equals(
   T  value     // karşılaştırmak için değer
   );
```

Parametreler

value

[in] Geçerli nesneyi karşılaştırmak için değer.

Dönen Değer

Nesneler eşitse true, aksi halde false döndürür.

HashCode

Geçerli nesne için hash kod değerini hesaplar.

```
int HashCode();
```

Dönen Değer

Hash kodu döndürür.

IComparable<T>

IComparable<T>, birinin diğerinden büyük, küçük veya eşit olup olmadığını bulmak için karşılaştırılabilecek nesneleri uygulamak için kullanılan bir arabirimdir

Tanım

IComparable <T> arabirimi, geçerli nesneyi aynı türün diğer bir nesne ile karşılaştırmak için bir yöntem tanımlar; bu nesnenin temelinde bu nesnelerin koleksiyonu sıralanabilir.

Deklarasyon

```
template<typename T>
interface IComparable : public IEqualityComparable<T>
```

Başlık

```
#include <Generic\Interfaces\IComparable.mqh>
```

Inheritance Hiyerarşisi

IEqualityComparable

IComparable

Direct descendants

CKeyValuePair

Sınıf Metodları

Metot	Tanım
Compare	Belirli değerdeki geçerli nesneyi karşılaştırır

Compare

Belirli değerdeki geçerli öğeyi karşılaştırır.

```
int Compare(
   T  value     // karşılaştırmak için değer
   );
```

Parametreler

value

[in] Geçerli nesneyi karşılaştırmak için değer.

Dönen Değer

Geçerli ve geçmiş nesne oranını ifade eden bir sayı döndürür:

- Eğer sonuç sıfırdan küçükse, geçerli nesne geçmiş bir nesneden daha küçüktür
- Eğer sonuç sıfırsa, geçerli nesne, geçmiş bir nesneye eşittir
- Eğer sonuç sıfırdan büyükse, geçerli nesne geçmiş bir nesneden daha büyüktür

IComparer<T>

IComparer <T>, biri diğerinden büyük, küçük veya eşit olan T türü iki nesneyi karşılaştıran genel bir sınıf uygulamak için bir arabirimdir

Tanım

IComparer<T> arabirimi, T nesnesinin bir koleksiyonunun sıralanabildiği temelinde T türündeki iki nesneyi karşılaştırmak için bir yöntem belirler.

Deklarasyon

```
template<typename T>
interface IComparer
```

Başlık

```
#include <Generic\Interfaces\IComparer.mqh>
```

Inheritance Hiyerarşisi

IKarşılaştırıcı

Direct descendants

CDefaultComparer

Sınıf Metodları

Metot	Tanım
Compare	T tipi iki değeri karşılaştırır.

Compare

T tipi iki değeri karşılaştırır.

```
int Compare(
   T  x,      // ilk değer
   T  y       // ikinci değer
   );
```

Parametreler

x

[in] Karşılaştırmak için ilk değer.

y

[in] Karşılaştırmak için ilk değer.

Dönen Değer

İki karşılaştırılan değerin oranını ifade eden bir sayı döndürür:

- Eğer sonuç sıfırdan küçükse, x küçüktür y den (x<y)
- Eğer sonuç sıfıra eşitse, x eşittir y ye (x=y)
- Eğer sonuç sıfırdan büyükse, x büyüktür y den (x>y)

IEqualityComparer<T>

IEqualityComparer<T>, T türündeki iki nesneyi karşılaştıran jenerik bir sınıf uygulamak için kullanılan bir arabirimdir.

Tanım

IEqualityComparer<T> arabirimi, T türü bir nesnenin karma kodunu almak ve T türü iki nesnenin eşit olup olmadığını kontrol etmek için yöntemleri tanımlar.

Deklarasyon

```
template<typename T>
interface IEqualityComparer
```

Başlık

```
#include <Generic\Interfaces\IEqualityComparer.mqh>
```

Inheritance Hiyerarşisi

IEqualityComparer

Direct descendants

CDefaultEqualityComparer

Sınıf Metodları

Metot	Tanım
Equals	T tipi iki değeri karşılaştırır.
HashCode	Hash kod değerini T türü nesneye dayalı olarak hesaplar

Equals

T tipi iki değeri karşılaştırır.

```
bool Equals(
   T  x,      // ilk değer
   T  y       // ikinci değer
   );
```

Parametreler

x

[in] Karşılaştırmak için ilk değer.

y

[in] Karşılaştırmak için ikinci değer.

Dönen Değer

Değerler eşitse true, aksi halde false döndürür.

HashCode

Hash kod değerini T türü nesneye dayalı olarak hesaplar.

```
int HashCode(
   T  value     // hesaplama için bir nesne
   );
```

Parametreler

value

[in] Hash kodu almak için istediğiniz bir nesne.

Dönen Değer

Hash kodu döndürür.

IList<T>

IList<T> genel veri listelerini uygulamak için bir arabirimdir.

Tanım

IList<T> arabirimi, indeksdeki bir öğeye erişmek, öğeleri aramak ve silmek, sıralama ve diğerlerini bulmak için listelerle çalışmak için temel yöntemleri tanımlar.

Deklarasyon

```
template<typename T>
interface IList : public ICollection<T>
```

Başlık

```
#include <Generic\Interfaces\IList.mqh>
```

Inheritance Hiyerarşisi

ICollection

IListe

Direct descendants

CArrayList

Sınıf Metodları

Metot	Tanım
TryGetValue	Belirtilen indekste bir liste öğesi alır
TrySetValue	Belirtilen indekste listeden bir değeri değiştirir
Insert	Belirtilen dizinde listeye bir öğe ekler
IndexOf	Listedeki bir değerin ilk ortaya çıkışını arar
LastIndexOf	Bir listede bir değerin son ortaya çıkışını arar
RemoveAt	Belirtilen indeksteki bir liste öğesini kaldırır

TryGetValue

Belirtilen indekste bir liste öğesini alır.

```
bool TryGetValue(
   const int  index,      // öğe indeksi
   T&         value       // yazmak için bir değişken
   );
```

Parametreler

index

[in] Listeden öğenin indeksi.

&value

[out] Listeden öğenin belirtilen değerinin yazılacağı değişken.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

TrySetValue

Belirtilen indeksteki listeden bir değeri değiştirir.

```
bool TrySetValue(
   const int  index,      // öğe indeksi
   T          value       // yeni değer
   );
```

Parametreler

index

[in] Listeden öğenin indeksi.

value

[in] Belirtilen liste öğesine atanacak yeni değer.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Insert

Belirtilen dizinde listeye bir öğe ekler.

```
bool Insert(
   const int  index,      // eklemek için dizin
   T          item        // eklenecek değer
   );
```

Parametreler

index

[in] Eklenecek dizin.

item

[in] Belirtilen dizine eklenecek değer.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

IndexOf

Listedeki bir değerin ilk ortaya çıkışını arar.

```
int IndexOf(
   T  item     // arama değeri
   );
```

Parametreler

item

[in] Aranan değer.

Dönen Değer

İlk bulunan öğenin dizinini döndürür. Eğer değer bulunmazsa -1 değeri döner.

LastIndexOf

Bir listede bir değerin son ortaya çıkışını arar.

```
int LastIndexOf(
   T  item     // arama değeri
   );
```

Parametreler

item

[in] Aranan değer.

Dönen Değer

En son bulunan öğenin dizinini döndürür. Eğer değer bulunmazsa -1 değeri döner.

RemoveAt

Belirtilen indekste bir liste öğesini kaldırır.

```
bool RemoveAt(
   const int  index     // öğe indeksi
   );
```

Parametreler

index

[in] Silmek istediğiniz öğenin indeksi.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

IMap<TKey, TValue>

IMap<TKey, TValue> anahtar/değer çiftlerinin jenerik koleksiyonlarını uygulamak için bir arabirimdir.

Tanım

The IMap<TKey, TValue> arabirimi verinin anahtar/değer çiftleri olarak depolandığı koleksiyonlarla ile çalışmak için temel metodları tanımlar.

Deklarasyon

```
template<typename TKey, typename TValue>
interface IMap : public ICollection<TKey>
```

Başlık

```
#include <Generic\Interfaces\IMap.mqh>
```

Inheritance Hiyerarşisi

ICollection

IHarita

Direct descendants

CHashMap, CSortedMap

Sınıf Metodları

Metot	Tanım
Add	Anahtar/değer çiftini bir koleksiyona ekler
Contains	Bir koleksiyonun belirli bir anahtar ile anahtar/değer tablosu içerip içermediğini belirler
Remove	Bir koleksiyondan bir anahtar/değer çiftinin ilk varlığını kaldırır
TryGetValue	Bir koleksiyondan belirli anahtar ile bir öğeyi alır
TrySetValue	Belirli anahtarda bir koleksiyondan anahtar/değer çiftinin değerini değiştirir
CopyTo	Belirtilen indeksten başlayarak, tüm anahtar/değer çiftlerini bir koleksiyondan belirtilen dizilere kopyalar

Add

Adds a key/value pair to a collection.

```
bool Add(
   TKey    key,       // anahtar
   TValue  value      // değer
   );
```

Parametreler

key

[in] Anahtar.

value

[in] Değer.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Contains

Birkoleksiyonun belirli bir anahtar ile anahtar/değer tablosu içerip içermediğini belirler.

```
bool Contains(
   TKey    key,       // anahtar
   TValue  value      // değer
   );
```

Parametreler

key

[in] Anahtar.

value

[in] Değer.

Dönen Değer

Eğer koleksiyon belirli bir anahtar ve değerle bir anahatar/değer çifti içeriyorsa true, aksi halde false döndürür

Remove

Bir koleksiyondanbir anahtar/değer çiftinin ilk varlığını kaldırır.

```
bool Remove(
   TKey  key      // anahtar
   );
```

Parametreler

key

[in] Anahtar.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

TryGetValue

Bir koleksiyondanbelirli anahtar ile bir öğeyi alır.

```
bool TryGetValue(
   TKey      key,       // anahtar
   TValue&   value      // değer yazmak için bir değişken
   );
```

Parametreler

key

[in] Anahtar.

&value

[out] anahtar/değer çiftinin belirtilen değeri yazılacak değişken.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

TrySetValue

Birkoleksiyondan anahtar/değer çiftinin değerini değiştirir.

```
bool TrySetValue(
   TKey    key,       // anahtar
   TValue  value      // yeni değer
   );
```

Parametreler

key

[in] Anahtar.

value

[in] Belirtilen anahtar/değer çiftine atanacak yeni değer.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

CopyTo

Bir koleksiyondan belirli bir indeksten başlayarak tüm anahtar/değer çiftlerini belirli dizilere kopyalar.

```
int CopyTo(
   TKey&       dst_keys[],       // anahtarlar yazmak için bir dizi
   TValue&     dst_values[],     // değerler yazmak için bir dizi
   const int   dst_start=0       // yazmak için başlangıç indeksi
   );
```

Parametreler

&dst_keys[]

[out] Koleksiyondaki tüm anahtarların yazılacağı bir dizi.

&dst_values[]

[out] Koleksiyondaki ilgili anahtarların değerlerinin yazılacağı bir dizi.

dst_start=0

[in] Kopyalamanın başladığı dizideki bir indeks.

Dönen Değer

Kopyalanan anahtar/değer çifti sayısını döndürür.

ISet<T>

ISet<T>, jenerik veri setlerini uygulamak için bir arabirimdir.

Tanım

ISet arabirimi, kümelerle çalışmak için temel yöntemleri tanımlar; örneğin kümelerin birleşimi ve kesişimi, strict ve strict olmayan alt kümelerin tanımı ve diğerleri.

Deklarasyon

```
template<typename T>
interface ISet : public ICollection<T>
```

Başlık

```
#include <Generic\Interfaces\ISet.mqh>
```

Inheritance Hiyerarşisi

ICollection

ISet

Direct descendants

CHashSet, CSortedSet

Sınıf Metodları

Metot	Tanım
ExceptWith	Geçerli koleksiyon ile geçmiş bir koleksiyon (dizi) arasındaki farkı üretir.
IntersectWith	Geçerli koleksiyonun kesişme işlemini ve geçmiş koleksiyonu (dizi) üretir.
SymmetricExceptWith	Geçerli koleksiyon ile geçmiş bir koleksiyon (dizi) arasındaki simetrik farkın işleyişini üretir.
UnionWith	Geçerli koleksiyonun birleşimini ve geçmiş bir koleksiyonu (dizi) üretir.
IsProperSubsetOf	Geçerli kümenin belirtilen koleksiyon veya dizinin uygun bir alt kümesi olup olmadığını belirler.
IsProperSupersetOf	Geçerli kümenin belirtilen koleksiyon veya dizinin uygun üst kümesi olup olmadığını belirler.
IsSubsetOf	Geçerli kümenin belirtilen koleksiyon veya dizinin alt kümesi olup olmadığını belirler.
IsSupersetOf	Geçerli kümenin belirtilen koleksiyon veya dizinin üst kümesi olup olmadığını belirler.

Metot	Tanım
Overlaps	Geçerli kümenin belirtilen koleksiyon veya dizeyle çakışıp çakmadığını belirler.
SetEquals	Geçerli kümenin, belirtilen koleksiyon veya dizinin tüm öğelerini içerip içermediğini belirler.

ExceptWith

Geçerli koleksiyon ile geçmiş bir koleksiyonun (dizi) arasındaki farkı üretir. Belirtilen koleksiyonda (dizi) bulunan tüm öğeleri geçerli koleksiyonundan (dizi) kaldırır.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
void ExceptWith(
   ICollection<T>*  collection     // koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
void ExceptWith(
   T&  array[]                     // dizi
   );
```

Parametreler

**collection*

[in] Geçerli kümeden hariç tutulacak bir koleksiyon.

&collection[]

[in] Geçerli kümeden hariç tutulacak bir dizi.

Not

Sonuç, geçerli koleksiyona (dizi) yazdırılır.

IntersectWith

Geçerli koleksiyonun kesişme işlemini ve geçmiş koleksiyonu (dizi) üretir. Geçerli koleksiyonu yalnızca belirtilen koleksiyonda (dizi) bulunan öğeleri içerecek şekilde değiştirir.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
void IntersectWith(
   ICollection<T>*  collection     // koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
void IntersectWith(
   T&  array[]                     // dizi
   );
```

Parametreler

**collection*

[in] Geçerli küme ile kesişecek bir koleksiyon.

&collection[]

[in] Geçerli küme ile kesişecek bir dizi.

Not

Sonuç, geçerli koleksiyona (dizi) yazdırılır.

SymmetricExceptWith

Geçerli koleksiyon ile geçmiş bir koleksiyon (dizi) arasındaki simetrik farkın operasyonunu üretir. Geçerli koleksiyonu, yalnızca kaynak nesnede veya belirtilen koleksiyonda (dizilim) bulunan öğeleri içerecek şekilde değiştirir, ancak her ikisini birden değiştirmez.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
void SymmetricExceptWith(
   T&  array[]                     // dizi
   );
```

Parametreler

**collection*

[in] Simetrik bir fark üretmek için bir koleksiyon.

&collection[]

[in] Simetrik bir fark üretmek için bir dizi.

Not

Sonuç, geçerli koleksiyona (dizi) yazdırılır.

UnionWith

Geçerli koleksiyonun ve geçmiş bir koleksiyonun (dizi) birleşimini üretir. Belirtilen koleksiyon (dizi) öğelerinden eksik olanı geçerli koleksiyona (dizi) ekler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
void UnionWith(
   ICollection<T>*  collection     // koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
void UnionWith(
   T&  array[]                     // dizi
   );
```

Parametreler

**collection*

[in] Mevcut kümenin birleşeceği bir koleksiyon.

&collection[]

[in] Geçerli kümenin birleşeceği bir dizi.

Not

Sonuç, geçerli koleksiyona (dizi) yazdırılır.

IsProperSubsetOf

Geçerli kümenin belirtilen koleksiyon veya dizinin uygun bir alt kümesi olup olmadığını belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // ilişkiyi belirlemek için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool IsProperSubsetOf(
   T&  array[]                     // ilişkiyi belirlemek için bir dizi
   );
```

Parametreler

**collection*

[in] İlişkiyi belirlemek için bir koleksiyon.

&collection[]

[in] İlişkiyi belirlemek için bir dizi.

Dönen Değer

Geçerli küme uygun bir alt küme ise true, aksi halde false döndürür.

IsProperSupersetOf

Geçerli kümenin belirtilen koleksiyon veya dizinin uygun üst kümesi olup olmadığını belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // ilişkiyi belirlemek için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool IsProperSupersetOf(
   T&  array[]                     // ilişkiyi belirlemek için bir dizi
   );
```

Parametreler

**collection*

[in] İlişkiyi belirlemek için bir koleksiyon.

&collection[]

[in] İlişkiyi belirlemek için bir dizi.

Dönen Değer

Eğer geçerli küme bir uygun üst küme ise true, değilse false döndürür.

IsSubsetOf

Geçerli kümenin belirtilen koleksiyon veya dizinin alt kümesi olup olmadığını belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // ilişkiyi belirlemek için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool IsSubsetOf(
   T&  array[]                     // ilişkiyi belirlemek için bir dizi
   );
```

Parametreler

**collection*

[in] İlişkiyi belirlemek için bir koleksiyon.

&collection[]

[in] İlişkiyi belirlemek için bir dizi.

Dönen Değer

Eğer geçerli küme bir alt küme ise true, aksi halde false döndürür.

IsSupersetOf

Geçerli kümenin belirtilen koleksiyon veya dizinin üst kümesi olup olmadığını belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // ilişkiyi belirlemek için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool IsSupersetOf(
   T&  array[]                     // ilişkiyi belirlemek için bir dizi
   );
```

Parametreler

**collection*

[in] İlişkiyi belirlemek için bir koleksiyon.

&collection[]

[in] İlişkiyi belirlemek için bir dizi.

Dönen Değer

Eğer geçerli küme bir üst küme ise true, aksi halde false döndürür.

Overlaps

Geçerli kümenin belirtilen koleksiyon veya dizeyle çakışıp çakmadığını belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool Overlaps(
   ICollection<T>*  collection     // karşılaştırmak için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool Overlaps(
   T&  array[]                     // karşılaştırmak için bir dizi
   );
```

Parametreler

**collection*

[in] Çakışmayı belirlemek için bir koleksiyon.

&collection[]

[in] Çakışmayı belirlemek için bir dizi.

Dönen Değer

Geçerli küme ve bir koleksiyon veya bir dizi örtüştüğünde true, aksi halde false döndürür.

SetEquals

Geçerli kümenin, belirtilen koleksiyon veya dizinin tüm öğelerini içerip içermediğini belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool SetEquals(
   ICollection<T>*  collection     // karşılaştırmak için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool SetEquals(
   T&  array[]                     // karşılaştırmak için bir dizi
   );
```

Parametreler

**collection*

[in] Öğeleri karşılaştırmak için bir koleksiyon.

&collection[]

[in] Öğeleri karşılaştırmak için bir koleksiyon.

Dönen Değer

Geçerli küme belirtilen koleksiyon veya dizinin tüm öğelerini içeriyorsa true, aksi halde false döndürür.

CDefaultComparer<T>

CDefaultComparer<T> Compare global metoduna dayanan IComparer <T> genel arayüzünü uygulayan bir yardımcı sınıftır.

Tanım

The CDefaultComparer<T> sınıfı, kullanıcı IComparer<T> arabirimini uygulayan başka bir sınıfı örtük olarak kullanmadıkça jenerik data koleksiyonlarında default olarak kullanılır.

Deklarasyon

```
template<typename T>
class CDefaultComparer : public IComparer<T>
```

Başlık

```
#include <Generic\Internal\DefaultComparer.mqh>
```

Inheritance Hiyerarşisi

IComparer

CDefaultComparer

Sınıf Metodları

Metot	Tanım
Compare	T tipi iki değeri karşılaştırır.

Compare

T tipi iki değeri karşılaştırır.

```
int Compare(
   T  x,      // ilk değer
   T  y       // ikinci değer
   );
```

Parametreler

x

[in] Karşılaştırmak için ilk değer.

y

[in] Karşılaştırmak için ikinci değer.

Dönen Değer

İki karşılaştırılan değerin oranını ifade eden bir sayı döndürür:

- Eğer sonuç sıfırdan küçükse, x küçüktür y den (x<y)
- Eğer sonuç sıfıra eşitse, x eşittir y ye (x=y)
- Eğer sonuç sıfırdan büyükse, x büyüktür y den (x>y)

Not

x ve y değerleri, T tipine bağlı olarak Compare golabal metodunun aşırı yüklerinden birine göre karşılaştırılır.

CDefaultEqualityComparer<T>

CDefaultEqualityComparer<T>, Equals<T> ve GetHashCode global metoduna dayanan IEqualityComparer<T> jenerik arayüzünü uygulayan yardımcı bir sınıftır.

Tanım

The CDefaultEqualityComparer<T> sınıfı, kullanıcı örtülü olarak IEqualityComparer<T> arabirimini uygulayan başka bir sınıf kullanmadıkça jenerik data koleksiyonlarında default olarak kullanılır.

Deklarasyon

```
template<typename T>
class CDefaultEqualityComparer : public IEqualityComparer<T>
```

Başlık

```
#include <Generic\Internal\DefaultEqualityComparer.mqh>
```

Inheritance Hiyerarşisi

IEqualityComparer

CDefaultEqualityComparer

Sınıf Metodları

Metot	Tanım
Equals	T tipi iki değeri karşılaştırır.
HashCode	Hash kod değerini T türü nesneye dayalı olarak hesaplar

Equals

T tipi iki değeri karşılaştırır.

```
bool Equals(
   T  x,      // ilk değer
   T  y       // ikinci değer
   );
```

Parametreler

x

[in] Karşılaştırmak için ilk değer.

y

[in] Karşılaştırmak için ikinci değer.

Dönen Değer

Değerler eşitse true, aksi halde false döndürür.

HashCode

Hash kod değerini T türü nesneye dayalı olarak hesaplar.

```
int HashCode(
   T  value     // hesaplama için bir nesne
   );
```

Parametreler

value

[in] Hash kodu almak için istediğiniz bir nesne.

Dönen Değer

Hash kodu döndürür.

CRedBlackTreeNode<T>

CRedBlackTreeNode<T>, CRedBlackTree<T> sınıfını uygulamada kullanılan yardımcı bir sınıftır.

Tanım

The CRedBlackTreeNode<T> sınıfı, CRedBlackTree<T>'ın bir düğümüdür. Ağaç navigation metodları sınıf içerisinde uygulanmaktadır.

Deklarasyon

```
template<typename T>
class CRedBlackTreeNode
```

Başlık

```
#include <Generic\RedBlackTree.mqh>
```

Sınıf Metodları

Metot	Tanım
Value	Bir düğüm değerini ayarlar ve döndürür
Parent	Üst düğüme bir işaretçi ayarlar ve döndürür
Left	Sol düğüme bir işaretçi ayarlar ve döndürür
Right	Sağ düğüme bir işaretçi ayarlar ve döndürür
Color	Düğüm rengini ayarlar ve döndürür
IsLeaf	Belitilen düğümün yaprak olup olmadığını belirler
CreateEmptyNode	Üst ve alt öğeler olmadan yeni bir siyah düğüm oluşturur ve ona bir işaretçi döndürür

Value (Get metodu)

Düğüm değerini döndürür.

```
T Value();
```

Dönen Değer

Düğüm değerini döndürür.

Value (Set metodu)

Düğüm değerini ayarlar

```
void Value(
   T  value     // düğüm değeri
   );
```

Parametreler

value

[in] Düğüm değeri.

Parent (Get metodu)

Üst düğüm için bir işaretçi döndürür.

```
CRedBlackTreeNode<T>* Parent();
```

Dönen Değer

Üst düğüm için bir işaretçi döndürür.

Parent (Set metodu)

Üst düğüm için bir işaretçi ayarlar.

```
void Parent(
   CRedBlackTreeNode<T>*  node     // üst düğüm için bir işaretçi
   );
```

Parametreler

node

[in] Üst düğüm için bir işaretçi.

Left (Get metodu)

Sol düğüme bir işaretçi döndürür.

```
CRedBlackTreeNode<T>* Left();
```

Dönen Değer

Sol düğüme bir işaretçi döndürür.

Left (Set metodu)

Sol düğüme bir işaretçi ayarlar.

```
void Left(
   CRedBlackTreeNode<T>*  node     // sol düğüm için bir işaretçi
   );
```

Parametreler

node

[in] Sol düğüm için bir işaretçi.

Right (Get metodu)

Sağ düğüm için bir işaretçi döndürür.

```
CRedBlackTreeNode<T>* Right();
```

Dönen Değer

Sağ düğüm için bir işaretçi döndürür.

Right (Set metodu)

Sağ düğüm için bir işaretçi ayarlar.

```
void Right(
   CRedBlackTreeNode<T>*  node     // sağ düğüm için bir işaretçi
   );
```

Parametreler

node

[in] Sağ düğüm için bir işaretçi.

Color (Get metodu)

Düğüm rengini döndürür.

```
ENUM_RED_BLACK_TREE_NODE_TYPE Color();
```

Dönen Değer

Düğüm rengini döndürür.

Color (Set metodu)

Düğüm rengini ayarlar.

```
void Color(
   ENUM_RED_BLACK_TREE_NODE_TYPE  clr     // düğüm rengi
   );
```

Parametreler

clr

[in] Düğüm rengi.

Not

Düğüm rengini ENUM_RED_BLACK_TREE_NODE_TYPE değerini kullanarak ayarlar. İki çeşit olabilir:

- RED_BLACK_TREE_NODE_RED — düğümün kırmzı rengi;
- RED_BLACK_TREE_NODE_BLACK — düğümün siyah rengi.

IsLeaf

Belirtilen düğümün bir yaprak olup olmadığını belirler.

```
bool IsLeaf();
```

Dönen Değer

Eğer düğüm bir yapraksa true, aksi halde false döndürür.

CreateEmptyNode

Üst ve alt öğeler olmadan yeni bir siyah düğüm oluşturur ve ona bir işaretçi döndürür.

```
static CRedBlackTreeNode<T>* CreateEmptyNode();
```

Dönen Değer

Yeni düğüme bir işaretçi döndürür.

CLinkedListNode<T>

CLinkedListNode<T>, CLinkedListNode<T> sınıfını uygulamada kullanılan yardımcı bir sınıftır.

Tanım

CLinkedListNode<T> sınıfı, çift bağlantılı CLinkedListNode<T> listesinin bir düğümüdür. Liste navigasyon metodları, sınıfın içerisinde uygulanmaktadır.

Deklarasyon

```
template<typename T>
class CLinkedListNode
```

Başlık

```
#include <Generic\LinkedList.mqh>
```

Sınıf Metodları

Metot	Tanım
List	CLinkedList<T> için bir işaretçi ayarlar ve döndürür
Next	Bir sonraki düğüm için bir işaretçi ayarlar ve döndürür
Previous	Önceki düğüm için bir işaretçi ayarlar ve döndürür
Value	Düğüm değerini ayarlar ve döndürür

List (Get metodu)

CLinkedList<T>'a bir işaretçi döndürür.

```
CLinkedList<T>* List();
```

Dönen Değer

CLinkedList<T> bağlantılı listesinebir işaretçi döndürür

List (Set metodu)

CLinkedList<T>'a bir işaretçi ayarlar.

```
void List(
   CLinkedList<T>*  value     // liste için bir işaretçi
   );
```

Parametreler

value

[in] CLinkedList<T>.bağlantılı listesi için bir işaretçi

Next (Get metodu)

Bir sonraki düğüm için bir işaretçi döndürür.

```
CLinkedListNode<T>* Next();
```

Dönen Değer

Bir sonraki düğüm için bir işaretçi döndürür.

Next (Set metodu)

Bir sonraki düğüm için bir işaretçi ayarlar

```
void Next(
   CLinkedListNode<T>*  value     // bir sonraki bir düğüm için bir işaretçi
   );
```

Parametreler

value

[in] Bir sonraki düğüm için bir işaretçi.

Previous (Get metodu)

Önceki düğüm için bir işaretçi döndürür.

```
CLinkedListNode<T>* Previous();
```

Dönen Değer

Önceki düğüm için bir işaretçi döndürür.

Previous (Set metodu)

Önceki düğüm için bir işaretçi ayarlar.

```
void Previous(
   CLinkedListNode<T>*  value     // önceki düğüm için bir işaretçi
   );
```

Parametreler

value

[in] Önceki düğüm için bir işaretçi.

Value (Get metodu)

Düğüm değerini döndürür.

```
T Value();
```

Dönen Değer

Düğüm değerini döndürür.

Value (Set metodu)

Düğüm değerini ayarlar

```
void Value(
   T  value     // Düğüm değeri
   );
```

Parametreler

value

[in] Düğüm değeri.

CKeyValuePair<TKey, TValue>

CKeyValuePair<TKey,TValue> sınıfı bir anahtar/değer çiftini uygular.

Tanım

CKeyValuePair<TKey,TValue> sınıfı anahtar/değer çiftinin anahtar ve değeriyle çalışmak için yöntemler uygular.

Deklarasyon

```
template<typename TKey, typename TValue>
class CKeyValuePair : public IComparable<CKeyValuePair<TKey,TValue>*>
```

Başlık

```
#include <Generic\HashMap.mqh>
```

Inheritance Hiyerarşisi

IEqualityComparable
  IComparable
    CKeyValuePair

Sınıf Metodları

Metot	Tanım
Key	Anahtar/değer çiftindeki anahtarını alır ve ayarlar
Value	Anahtar/değer çiftindeki değeri alır ve ayarlar
Clone	Anahtar ve değeri şu ankine eşit olan yeni bir anahtar/değer çifti oluşturur
Compare	Geçerli anahtar/değer çifti ile belirli bir çifti karşılaştırır
Equals	Geçerli anahtar/değer çiftinin ve belirtilen çiftin eşit olup olmadığını kontrol eder
HashCode	Anahtar/değer çiftini baz alarak hash değeri hesaplar

Key (Get metodu)

Anahtar/değer çifti içerisindeki anahtarı alır.

```
TKey Key();
```

Dönen Değer

Anahtarı döndürür.

Key (Set metodu)

Anahtar/değer çifti içersindeki anahtarı ayarlar.

```
void Key(
   TKey  key     // anahtar
   );
```

Parametreler

key

[in] Anahtar.

Value (Get metodu)

Anahtar/değer çiftindeki değeri alır.

```
TValue Value();
```

Dönen Değer

Değeri döndürür.

Value (Set metodu)

Anahtar/değer çiftindeki değeri ayarlar.

```
void Value(
   TValue  value     // değer
   );
```

Parametreler

value

[in] Değer

Clone

Anahtar ve değeri şimdiki çifte eşit olan yeni bir anahtar/değer çifti oluşturur.

```
TValue>* Clone();
```

Dönen Değer

Yeni bir anahtar/değer döndürür

Compare

Geçerli anahtar/değer çifti ile belirtilen çifti karşılaştırır.

```
int Compare(
   CKeyValuePair<TKeyTValue>*  pair     // karşılaştırmak için çift
   );
```

Parametreler

pair

[in] Karşılaştırmak için çift.

Dönen Değer

Geçerli ve geçmiş anahtar/değer çiftlerinin oranını ifade eden bir sayı döndürür:

- Sonuç sıfırdan küçükse, geçerli anahtar/değer çifti geçmiş olandan düşüktür
- Sonuç sıfırsa, geçerli anahtar/değer çifti geçmiş olan çifte eşittir
- Sonuç sıfırdan büyükse, geçerli anahtar/değer çifti geçmiş olandan daha büyüktür

Not

Anahtar/değer çiftleri onların anahtarları baz alınarak karşılaştırılır.

Equals

Geçerli anahtar/değer çiftinin ve belirtilen çiftin eşit olup olmadığını kontrol eder.

```
bool Equals(
   CKeyValuePair<TKeyTValue>*  pair     // karşılaştırmak için çift
   );
```

Parametreler

pair

[in] Karşılaştırmak için çift

Dönen Değer

Eğer anahtar/değer çifti eşitse true, aksi halde false döndürür.

Not

Anahtar/değer çiftleri onların anahtarları baz alınarak karşılaştırılır.

HashCode

Anahtar/değer çifti baz alınarak hash değeri hesaplar.

```
int HashCode();
```

Dönen Değer

Hash kodu döndürür.

Not

Anahtar/değer çiftinin hash kodu anahtar hash koda eşittir.

CArrayList<T>

CArrayList<T> IList <T> arabirimini uygulayan jenerik bir sınıfdır.

Tanım

The CArrayList<T> sınıfı, T türü dinamik veri listesinin bir uygulamasıdır. Bu sınıf, bir elemana endeksle erişmek, elemanları aramak ve silmek, sıralama ve diğerlerini yapmak için listeyle çalışmanın temel yöntemlerini sağlar.

Deklarasyon

```
template<typename T>
class CArrayList : public IList<T>
```

Başlık

```
#include <Generic\ArrayList.mqh>
```

Inheritance Hiyerarşisi

ICollection
  IList
    CArrayList

Sınıf Metodları

Metot	Tanım
Capacity	Bir listenin geçerli kapasitesini alır ve ayarlar
Count	Listedeki öğelerin sayısını döndürür
Contains	Bir listede belirtilen değere sahip bir öğe olup olmadığını belirler
TrimExcess	Öğelerin gerçek sayısı için bir listenin kapasitesini ayarlar
TryGetValue	Belirtilen dizinde listenin bir öğesini döndürür
TrySetValue	Belirtilen dizindeki liste öğesinin değerini ayarlar.
Add	Listeye bir öğe ekler
AddRange	Listeye bir koleksiyon veya öğelerin bir dizisini ekler
Insert	Belirtilen dizinde listeye bir öğe ekler
InsertRange	Belirtilen dizindeki listeye bir koleksiyon veya öğelerin bir dizisini ekler.
CopyTo	Bir listenin tüm öğelerini, belirtilen dizinde başlayan belirtilen diziye kopyalar
BinarySearch	Artan sıralı listede belirtilen değere göre arama yapar
IndexOf	Listedeki bir değerin ilk ortaya çıkışını arar

Metot	Tanım
LastIndexOf	Bir listede bir değerin son ortaya çıkışını arar
Clear	Koleksiyondaki tüm öğeleri kaldırır
Remove	Belirtilen öğenin ilk bulunduğu öğeyi listeden kaldırır
RemoveAt	Listenin belirtilen dizinindeki bir öğeyi kaldırır.
RemoveRange	Listeden bir dizi öğeyi kaldırır
Reverse	Listedeki öğelerin sıralamasını tersine çevirir
Sort	Listedeki elemanları sıralar

Capacity (Get metodu)

Geçerli liste kapasitesini döndürür.

```
int Capacity();
```

Dönen Değer

Geçerli liste kapasitesini döndürür.

Capacity (Set Metodu)

Bir listenin geçerli kapasitesini ayarlar.

```
void Capacity(
   const int  capacity     // kapasite değeri
   );
```

Parametreler

capacity

[in] Kapasitenin yeni bir değeri.

Count

Listedeki öğelerin sayısını döndürür.

```
int Count();
```

Dönen Değer

Öğelerin sayısını döndürür.

Contains

Listede belirtilen değere sahip bir öğe olup olmadığını belirler.

```
bool Contains(
   T  item     // arama değeri
   );
```

Parametreler

item

[in] Aranan değer.

Dönen Değer

Belirtilen değere sahip bir öğe listede bulunursa true, aksi halde false döndürür.

TrimExcess

Bir listenin kapasitesini öğelerin gerçek sayısına ayarlar ve böylece kullanılmayan belleği boşa çıkarır.

```
void TrimExcess();
```

TryGetValue

Belirtilen dizinde listenin bir öğesini döndürür.

```
bool TryGetValue(
   const int  index,      // indeks
   T&         value       // yazmak için bir değişken
   );
```

Parametreler

index

[in] Almak istediğiniz liste öğesi değerinin indeksi.

&value

[out] Öğenin değerini yazacak değişken.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

TrySetValue

Belirtilen dizindeki liste öğesinin değerini ayarlar.

```
bool TrySetValue(
   const int  index,      // indeks
   T          value       // öğenin değeri
   );
```

Parametreler

index

[in] Liste öğesinin değerini belirlemek istediğiniz dizin.

value

[in] Liste öğesinin değerini ayarlar.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Add

Listeye bir öğe ekler.

```
bool Add(
   T  value     // öğenin değeri
   );
```

Parametreler

value

[in] Eklenecek öğenin değeri

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

AddRange

Listeye bir koleksiyon veya öğelerin bir dizisini ekler.

Bir dizi ekleyen versiyon.

```
bool AddRange(
   const T&  array[]               // eklenecek dizi
   );
```

Bir koleksiyon ekleyen versiyon.

```
bool AddRange(
   ICollection<T>*  collection     // eklenecek bir koleksiyon
   );
```

Parametreler

&array[]

[in] Eklenecek bir dizi.

**collection*

[in] Eklenecek bir koleksiyon.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Insert

Belirtilen dizinde listeye bir öğe ekler.

```
bool Insert(
   const int  index,      // eklemek için dizin
   T          item        // eklenecek değer
   );
```

Parametreler

index

[in] Eklenecek dizin.

item

[in] Belirtilen dizine eklenecek değer.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

InsertRange

Belirtilen dizindeki listeye bir koleksiyon veya öğelerin bir dizisini ekler.

Bir diziyi ekleyen versiyon.

```
bool InsertRange(
   const int  index,               // eklemek için dizin
   const T&   array[]              // eklenecek dizin
   );
```

Bir koleksiyon ekleyen versiyon.

```
bool InsertRange(
   const int        index,         // eklemek için dizin
   ICollection<T>*  collection     // eklenecek bir koleksiyon
   );
```

Parametreler

index

[in] Eklenecek dizin.

&array[]

[in] Belirtilen dizine eklenecek bir dizi.

**collection*

[in] Belirtilen dizine eklenecek bir koleksiyon.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

CopyTo

Bir listenin tüm öğelerini, belirtilen dizinde başlayan belirtilen diziye kopyalar.

```
int CopyTo(
   T&          dst_array[],     // yazı için bir dizi
   const int   dst_start=0      // yazmak için başlangıç indeksi
   );
```

Parametreler

&dst_array[]

[out] Listenin öğelerinin yazılacağı bir dizi.

dst_start=0

[in] Kopyalamanın başladığı dizideki bir indeks.

Dönen Değer

Kopyalanan öğelerin sayısını döndürür.

BinarySearch

Artan sıralı listede belirtilen değere göre arama yapar.

Öğeleri karşılaştırmak için IComparable<T> arayüzünü uygulayan sınıfı kullanarak belirtilen değer aralığında arama yapan versiyon.

```
int BinarySearch(
   const int       index,        // başlangıç indeksi
   const int       count,        // arama aralığı
   T               item,         // arama değeri
   IComparer<T>*   comparer      // karşılaştırmak için arayüz
   );
```

Öğeleri karşılaştırmak için IComparable<T> arayüzünü uygulayan sınıfı kullanarak arama yapan versiyon.

```
int BinarySearch(
   T               item,         // arama değeri
   IComparer<T>*   comparer      // karşılaştırmak için arayüz
   );
```

Öğeleri karşılaştırmak için ::Compare global metoduu kullanarak arama yapan versiyon

```
int BinarySearch(
   T  item                       // arama değeri
   );
```

Parametreler

index

[in] Aramanın başladığı başlangıç dizisi.

count

[in] Arama aralığının uzunluğu

item

[in] Aranan değer.

**comparer*

[in] Bir arayüz öğeleri karşılaştırmak için.

Dönen Değer

Bulunan öğrenin dizinini döndürür. Arama değeri bulunamazsa, değeri en yakın olan en küçük öğenin dizinini döndürür.

IndexOf

Listedeki bir değerin ilk ortaya çıkışını arar.

Listenin tamamında arama yapan versiyon.

```
int IndexOf(
   T  item                    // arama değeri
   );
```

Belirtilen konumdan listenin sonuna kadar arama yapan versiyon.

```
int IndexOf(
   T          item,           // arama değeri
   const int  start_index     // the starting index
   );
```

Belirtilen aralıktaki belirtilen konumdan arama yapan versiyon.

```
int IndexOf(
   T          item,           // arama değeri
   const int  start_index,    // başlangıç indeksi
   const int  count           // arama aralığı
   );
```

Parametreler

item

[in] Aranan değer.

start_index

[in] Aramanın başladığı başlangıç dizisi.

count

[in] Arama aralığının uzunluğu

Dönen Değer

İlk bulunan öğenin dizinini döndürür. Eğer değer bulunmazsa -1 değeri döner.

LastIndexOf

Bir listede bir değerin son ortaya çıkışını arar.

Listenin tamamında arama yapan versiyon.

```
int LastIndexOf(
   T  item                    // arama değeri
   );
```

Belirtilen konumdan listenin sonuna kadar arama yapan versiyon.

```
int LastIndexOf(
   T           item,          // arama değeri
   const int   start_index    // başlangıç indeksi
   );
```

Belirtilen aralıktaki belirtilen konumdan arama yapan versiyon.

```
int LastIndexOf(
   T           item,          // arama değeri
   const int   start_index,   // başlangıç indeksi
   const int   count          // arama aralığı
   );
```

Parametreler

item

[in] Aranan değer.

start_index

[in] Aramanın başladığı başlangıç dizisi.

count

[in] Arama aralığının uzunluğu

Dönen Değer

En son bulunan öğenin dizinini döndürür. Eğer değer bulunmazsa -1 değeri döner.

Clear

Bir koleksiyonun tüm öğelerini kaldırır.

```
void Clear();
```

Remove

Belirtilen öğenin ilk bulunduğu öğeyi listeden kaldırır.

```
bool Remove(
   T  item     // öğe değeri
   );
```

Parametreler

item

[in] Silinecek öğenin değeri.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

RemoveAt

Listenin belirtilen dizinindeki bir öğeyi kaldırır.

```
bool RemoveAt(
   const int  index     // indeks
   );
```

Parametreler

index

[in] Kaldırılacak elemanın indeksi.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

RemoveRange

Listeden bir dizi öğeyi kaldırır.

```
bool RemoveRange(
   const int  start_index,     // başlangıç indeksi
   const int  count            // öğelerin sayısı
   );
```

Parametreler

start_index

[in] Silme işleminin başladığı başlangıç dizini.

count

[in] Silinecek öğe sayısı

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Reverse

Listedeki öğelerin sıralamasını tersine çevirir.

Tüm listeyle çalışmak için kullanılan versiyon.

```
bool Reverse();
```

Belirtilen liste öğesi aralığında çalışma versiyonu.

```
bool Reverse(
   const int  start_index,     // başlangıç indeksi
   const int  count            // öğelerin sayısı
   );
```

Parametreler

start_index

[in] Başlangıç indeksi.

count

[in] Operasyona katılan liste öğelerinin sayısı.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Sort

Listedeki öğeleri sıralar.

Listedeki tüm öğeleri sıralayan versiyon.

```
bool Sort();
```

Öğeleri karşılaştırmak için IComparable<T> arabirimini uygulayan sınıfı kullanarak listedeki tüm öğeleri sıralayan versiyon.

```
bool Sort(
   IComparer<T>*  comparer         // karşılaştırmak için arayüz
   );
```

Öğeleri karşılaştırmak için IComparable<T> arayüzünü uygulayan sınıfı kullanarak listedeki öğelerin belirtilen aralığını sıralayan versiyon.

```
bool Sort(
   const int  start_index,     // başlangıç indeksi
   const int  count            // öğelerin sayısı
   IComparer<T>*  comparer     // karşılaştırmak için arayüz
   );
```

Parametreler

**comparer*

[in] Bir arayüz öğeleri karşılaştırmak için.

start_index

[in] Sıralama işleminin başladığı başlangıç dizini.

count

[in] Sıralama aralığının uzunluğu.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

CHashMap<TKey, TValue>

CHashMap<TKey, TValue>, IMap<TKey, TValue> arayüzünü uygulayan jenerik bir sınıftır.

Tanım

CHashMap <TKey, TValue> sınıfı, dinamik hash tablosunun bir uygulamasıdır; veriler, sıralı olmayan anahtar/değer çiftleri biçiminde, anahtar tekliği gereksinimini dikkate alarak depolanır. Bu sınıf, anahtar/değer çifti aramak ve silmek için anahtara bir değer girmek, diğer bir anahtar/değer çifti aramak ve bunlara erişmek gibi bir hash tablo ile çalışmak için temel yöntemler sağlar.

Deklarasyon

```
template<typename TKey, typename TValue>
class CHashMap : public IMap<TKey, TValue>
```

Başlık

```
#include <Generic\HashMap.mqh>
```

Inheritance Hiyerarşisi

ICollection
  IMap
    CHashMap

Sınıf Metodları

Metot	Tanım
Add	Hash tabloya bir anahtar/değer çifti ekler
Count	Hash tablodaki öğe sayısını döndürür
Comparer	Hash tabloyu düzenlemek için kullanılan IEqualityComparer<T> arabirimine işaretçi döndürür
Contains	Hash tabloya belirtilen anahtar/değer çifti içerilip içermediğini belirler
ContainsKey	Hash tablosunun belirtilen anahtarla anahtar/değer çiftini içerip içermediğini belirler.
ContainsValue	CHashMap<TKey, TValue>, IMap<TKey, TValue> arabirimini uygulayan jenerik bir sınıftır
CopyTo	Belirtilen dizinden başlayarak, hash tablosundaki tüm anahtar/değer çiftlerini belirtilen dizilere kopyalar
Clear	Hash tablodaki tüm öğeleri kaldırır
Remove	Anahtar/değer çiftinin ilk örneğini hash tablosundan kaldırır.
TryGetValue	Hash tablosundan belirtilen anahtarla bir öğeyi alır

Metot	Tanım
TrySetValue	Belirtilen anahtarda hash tablosundan bir anahtar/değer çifti değerini değiştirir

Add

Hash tablosuna bir anahtar/değer çifti ekler.

Oluşturulan anahtar/değer çiftini ekleyen bir versiyon.

```
bool Add(
   CKeyValuePair<TKeyTValue>*  pair     // anahtar/değer çifti
   );
```

Belirtilen anahtar ve değer ile yeni bir anahtar/değer çifti ekleyen bir versiyon.

```
bool Add(
   TKey    key,                         // anahtar
   TValue  value                        // değer
   );
```

Parametreler

**pair*

[in] anahtar/değer çifti.

key

[in] Anahtar.

value

[in] Değer.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Count

Hash tablosundaki öğe sayısını döndürür.

```
int Count();
```

Comparer

Hash tablosunu düzenlemek için kullanılan IEqualityComparer<T> arabirimine işaretçi döndürür.

```
IEqualityComparer<TKey>* Comparer() const;
```

Dönen Değer

IEqualityComparer<T> arabirimine işaretçi döndürür.

Contains

Hash tablosunda belirli anahtar/değer çiftinin bulunup bulunmadığınıbelirler.

Oluşturulan anahtar/değer çifti ile çalışma versiyonu.

```
bool Contains(
   CKeyValuePair<TKeyTValue>*  item     // anahtar/değer çifti
   );
```

Ayrı olarak ayarlanmış bir anahtar ve değer biçiminde bir anahtar/değer çifti ile çalışma versiyonu.

```
bool Contains(
   TKey   key,                          // anahtar
   TValue value                         // değer
   );
```

Parametreler

**item*

[in] anahtar/değer çifti.

key

[in] Anahtar.

value

[in] Değer.

Dönen Değer

Hash tablosunda belirtilen anahtar ve değer ile anahtar/değer çifti varsa true, aksi halde false döndürür.

ContainsKey

Hash tablosunun belirli bir anahtarda anahtar/değer çiftiiçerip içermediğini belirler.

```
bool ContainsKey(
   TKey  key     // anahtar
   );
```

Parametreler

key

[in] Anahtar.

Dönen Değer

Hash tablosunda belirtilen anahtarla bir anahtar/değer çifti varsa true değerini, aksi halde false değerini döndürür.

ContainsValue

Hash tablosunun belirli bir değerdeanahtar/değer çiftiiçerip içermediğini belirler.

```
bool ContainsValue(
   TValue  value     // değer
   );
```

Parametreler

value

[in] Değer.

Dönen Değer

Hash tablosunda belirtilen değere sahip bir anahtar/değer çifti varsa true, aksi halde false döndürür.

CopyTo

Hash tablosundan tüm anahtar/değer çiftlerinibelirli bir indeksten başlayarak belirli dizilere kopyalar.

Hash tabloyu anahtar/değer çifti dizisine kopyalar.

```
int CopyTo(
   CKeyValuePair<TKeyTValue>*&  dst_array[],     // anahtar/değer çifti yazmak için bi
   const int                    dst_start=0      // yazmak için başlangıç indeksi
   );
```

Anahtar ve değerler için dizileri ayırmak amacıyla bir hash tablosunu kopyalayan versiyon.

```
int CopyTo(
   TKey&       dst_keys[],                       // anahtarlar yazmak için bir dizi
   TValue&     dst_values[],                     // değerler yazmak için bir dizi
   const int   dst_start=0                       // yazmak için başlangıç indeksi
   );
```

Parametreler

&dst_array[]

[out] Hash tablosundan tüm çiftlerin yazılacağı bir dizi.

&dst_keys[]

[out] Hash tablosundan tüm anahtarların yazılacağı bir dizi.

&dst_values[]

[out] Hash tablosundan tüm değerlerin yazılacağı bir dizi.

dst_start=0

[in] Kopyalamanın başladığı dizin indeksi.

Dönen Değer

Kopyalanan anahtar/değer çifti sayısını döndürür.

Clear

Hash tablosundan tüm öğeleri kaldırır.

```
void Clear();
```

Remove

anahtar/değer çiftinin ilk örneğini hash tablosundankaldırır.

Oluşturulan anahtar/değer çifti temel alınarak bir anahtar/değer çifti kaldıran versiyon.

```
bool Remove(
   CKeyValuePair<TKeyTValue>*  item     // anahtar değer çifti"
   );
```

Anahtara dayalı bir anahtar/değer çifti kaldıran versiyon.

```
bool Remove(
   TKey  key                           // anahtar
   );
```

Parametreler

**item*

[in] anahtar/değer çifti.

key

[in] Anahtar.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

TryGetValue

Belirtilen bir anahtarlahash tablosundanbir öğe alır.

```
bool TryGetValue(
   TKey     key,       // anahtar
   TValue&  value      // değer yazmak için bir değişken
   );
```

Parametreler

key

[in] Anahtar.

&value

[out] anahtar/değer çiftinin belirtilen değeri yazılacak değişken.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

TrySetValue

Belirtilen anahtarda hash tablosundan bir anahtar/değer çiftinin değerini değiştirir.

```
bool TrySetValue(
   TKey    key,       // anahtar
   TValue  value      // yeni değer
   );
```

Parametreler

key

[in] Anahtar.

value

[in] Belirtilen anahtar/değer çiftine atanacak yeni değer.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

CHashSet<T>

CHashSet<T>, ISet<T> arabirimini uygulayan jenerik bir sınıfdır.

Tanım

CHashSet<T> sınıfı, T türü sırasız dinamik veri kümesinin bir uygulamasıdır ve her değerin tekli olması gerekir. Bu sınıf, kümeler ve ilgili işlemlerle çalışmak için temel yöntemler sağlar; örneğin kümelerin birleşimi ve kesişimi, strict ve strict olmayan alt kümelerin tanımı ve diğerleri.

Deklarasyon

```
template<typename T>
class CHashSet : public ISet<T>
```

Başlık

```
#include <Generic\HashSet.mqh>
```

Inheritance Hiyerarşisi

ICollection
  ISet
    CHashSet

Sınıf Metodları

Metot	Tanım
Add	Bir kümeye bir öğe ekler
Count	Bir kümedeki öğe sayısını döndürür
Comparer	Bir kümede belirtilen değere sahip bir öğe olup olmadığını belirler
Contains	Bir kümeyi düzenlemek için kullanılan IEqualityComparer<T> arabirimine bir işaretçi döndürür
TrimExcess	Setin kapasitesini gerçek öğe sayısına ayarlar ve böylece kullanılmayan belleği boşa çıkarır.
CopyTo	Bir kümedeki tüm öğeleri, belirtilen dizinden başlayarak belirtilen diziye kopyalar
Clear	Kümedeki tüm öğeleri kaldırır
Remove	Belirtilen öğeyi bir kümeden kaldırır.
ExceptWith	Geçerli koleksiyon ile geçmiş bir koleksiyon (dizi) arasındaki farkı üretir.
IntersectWith	Geçerli koleksiyonun kesişme işlemini ve geçmiş koleksiyonu (dizi) üretir.

Metot	Tanım
SymmetricExceptWith	Geçerli koleksiyon ile geçmiş bir koleksiyon (dizi) arasındaki simetrik farkın işleyişini üretir.
UnionWith	Geçerli koleksiyonun birleşimini ve geçmiş bir koleksiyonu (dizi) üretir.
IsProperSubsetOf	Geçerli kümenin belirtilen koleksiyon veya dizinin uygun bir alt kümesi olup olmadığını belirler.
IsProperSupersetOf	Geçerli kümenin belirtilen koleksiyon veya dizinin uygun üst kümesi olup olmadığını belirler.
IsSubsetOf	Geçerli kümenin belirtilen koleksiyon veya dizinin alt kümesi olup olmadığını belirler.
IsSupersetOf	Geçerli kümenin belirtilen koleksiyon veya dizinin üst kümesi olup olmadığını belirler.
Overlaps	Geçerli kümenin belirtilen koleksiyon veya dizeyle çakışıp çakmadığını belirler.
SetEquals	Geçerli kümenin, belirtilen koleksiyon veya dizinin tüm öğelerini içerip içermediğini belirler.

Add

Bir kümeye bir öğe ekler.

```
bool Add(
   T  value     // öğenin değeri
   );
```

Parametreler

value

[in] Eklenecek öğenin değeri

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Count

Bir kümedeki öğe sayısını döndürür.

```
int Count();
```

Dönen Değer

Öğelerin sayısını döndürür.

Contains

Bir kümede belirtilen değere sahip bir öğe olup olmadığını belirler.

```
bool Contains(
   T  item     // arama değeri
   );
```

Parametreler

item

[in] Aranan değer.

Dönen Değer

Belirtilen değere sahip bir öğe kümede bulunursa true, aksi halde false döndürür.

Comparer

Bir kümeyi düzenlemek için kullanılan IEqualityComparer<T> arabirimine bir işaretçi döndürür.

```
IEqualityComparer<T>* Comparer() const;
```

Dönen Değer

IEqualityComparer<T> arabirimine işaretçi döndürür.

TrimExcess

Bir setin kapasitesini gerçek öğe sayısına ayarlar ve böylece kullanılmayan belleği boşa çıkarır.

```
void TrimExcess();
```

CopyTo

Bir kümedeki tüm öğeleri, belirtilen dizinden başlayarak belirtilen diziye kopyalar.

```
int CopyTo(
   T&          dst_array[],      // yazmak için bir dizi
   const int   dst_start=0       // yazmak için başlangıç indeksi
   );
```

Parametreler

&dst_array[]

[out] Kümenin elemanlarının yazılacağı bir dizi.

dst_start=0

[in] Kopyalamanın başladığı dizideki bir indeks.

Dönen Değer

Kopyalanan öğelerin sayısını döndürür.

Clear

Bir setten tüm öğeleri kaldırır.

```
void Clear();
```

Remove

Belirtilen öğeyi bir kümeden kaldırır.

```
bool Remove(
   T  item     // öğe değeri
   );
```

Parametreler

item

[in] Silinecek öğenin değeri.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

ExceptWith

Geçerli koleksiyon ile geçmiş bir koleksiyonun (dizi) arasındaki farkı üretir. Belirtilen koleksiyonda (dizi) bulunan tüm öğeleri geçerli koleksiyonundan (dizi) kaldırır.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
void ExceptWith(
   ICollection<T>*  collection     // koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
void ExceptWith(
   T&  array[]                     // dizi
   );
```

Parametreler

**collection*

[in] Geçerli kümeden hariç tutulacak bir koleksiyon.

&collection[]

[in] Geçerli kümeden hariç tutulacak bir dizi.

Not

Sonuç, geçerli koleksiyona (dizi) yazdırılır.

IntersectWith

Geçerli koleksiyonun kesişme işlemini ve geçmiş koleksiyonu (dizi) üretir. Geçerli koleksiyonu yalnızca belirtilen koleksiyonda (dizi) bulunan öğeleri içerecek şekilde değiştirir.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
void IntersectWith(
   ICollection<T>*  collection     // koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
void IntersectWith(
   T&  array[]                     // dizi
   );
```

Parametreler

**collection*

[in] Geçerli küme ile kesişecek bir koleksiyon.

&collection[]

[in] Geçerli küme ile kesişecek bir dizi.

Not

Sonuç, geçerli koleksiyona (dizi) yazdırılır.

SymmetricExceptWith

Geçerli koleksiyon ile geçmiş bir koleksiyon (dizi) arasındaki simetrik farkın operasyonunu üretir. Geçerli koleksiyonu, yalnızca kaynak nesnede veya belirtilen koleksiyonda (dizilim) bulunan öğeleri içerecek şekilde değiştirir, ancak her ikisini birden değiştirmez.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
void SymmetricExceptWith(
   T&  array[]                     // dizi
   );
```

Parametreler

**collection*

[in] Simetrik bir fark üretmek için bir koleksiyon.

&collection[]

[in] Simetrik bir fark üretmek için bir dizi.

Not

Sonuç, geçerli koleksiyona (dizi) yazdırılır.

UnionWith

Geçerli koleksiyonun ve geçmiş bir koleksiyonun (dizi) birleşimini üretir. Belirtilen koleksiyon (dizi) öğelerinden eksik olanı geçerli koleksiyona (dizi) ekler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
void UnionWith(
   ICollection<T>*  collection     // koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
void UnionWith(
   T&  array[]                     // dizi
   );
```

Parametreler

**collection*

[in] Mevcut kümenin birleşeceği bir koleksiyon.

&collection[]

[in] Geçerli kümenin birleşeceği bir dizi.

Not

Sonuç, geçerli koleksiyona (dizi) yazdırılır.

IsProperSubsetOf

Geçerli kümenin belirtilen koleksiyon veya dizinin uygun bir alt kümesi olup olmadığını belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // ilişkiyi belirlemek için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool IsProperSubsetOf(
   T&  array[]                     // ilişkiyi belirlemek için bir dizi
   );
```

Parametreler

**collection*

[in] İlişkiyi belirlemek için bir koleksiyon.

&collection[]

[in] İlişkiyi belirlemek için bir dizi.

Dönen Değer

Geçerli küme uygun bir alt küme ise true, aksi halde false döndürür.

IsProperSupersetOf

Geçerli kümenin belirtilen koleksiyon veya dizinin uygun üst kümesi olup olmadığını belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // ilişkiyi belirlemek için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool IsProperSupersetOf(
   T&  array[]                     // ilişkiyi belirlemek için bir dizi
   );
```

Parametreler

**collection*

[in] İlişkiyi belirlemek için bir koleksiyon.

&collection[]

[in] İlişkiyi belirlemek için bir dizi.

Dönen Değer

Eğer geçerli küme bir uygun üst küme ise true, değilse false döndürür.

IsSubsetOf

Geçerli kümenin belirtilen koleksiyon veya dizinin alt kümesi olup olmadığını belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // ilişkiyi belirlemek için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool IsSubsetOf(
   T&  array[]                     // ilişkiyi belirlemek için bir dizi
   );
```

Parametreler

**collection*

[in] İlişkiyi belirlemek için bir koleksiyon.

&collection[]

[in] İlişkiyi belirlemek için bir dizi.

Dönen Değer

Eğer geçerli küme bir alt küme ise true, aksi halde false döndürür.

IsSupersetOf

Geçerli kümenin belirtilen koleksiyon veya dizinin üst kümesi olup olmadığını belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // ilişkiyi belirlemek için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool IsSupersetOf(
   T&  array[]                     // ilişkiyi belirlemek için bir dizi
   );
```

Parametreler

**collection*

[in] İlişkiyi belirlemek için bir koleksiyon.

&collection[]

[in] İlişkiyi belirlemek için bir dizi.

Dönen Değer

Eğer geçerli küme bir üst küme ise true, aksi halde false döndürür.

Overlaps

Geçerli kümenin belirtilen koleksiyon veya dizeyle çakışıp çakmadığını belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool Overlaps(
   ICollection<T>*  collection     // karşılaştırmak için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool Overlaps(
   T&  array[]                     // karşılaştırmak için bir dizi
   );
```

Parametreler

**collection*

[in] Çakışmayı belirlemek için bir koleksiyon.

&collection[]

[in] Çakışmayı belirlemek için bir dizi.

Dönen Değer

Geçerli küme ve bir koleksiyon veya bir dizi örtüştüğünde true, aksi halde false döndürür.

SetEquals

Geçerli kümenin, belirtilen koleksiyon veya dizinin tüm öğelerini içerip içermediğini belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool SetEquals(
   ICollection<T>*  collection     // karşılaştırmak için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool SetEquals(
   T&  array[]                     // karşılaştırmak için bir dizi
   );
```

Parametreler

**collection*

[in] Öğeleri karşılaştırmak için bir koleksiyon.

&collection[]

[in] Öğeleri karşılaştırmak için bir dizi.

Dönen Değer

Geçerli küme belirtilen koleksiyon veya dizinin tüm öğelerini içeriyorsa true, aksi halde false döndürür.

CLinkedList<T>

CLinkedList<T>, ICollection<T> arabirimini uygulayan bir jenerik sınıftır.

Tanım

CLinkedList<T> sınıfı, T tipi dinamik çift bağlantılı veri listesinin bir uygulamasıdır. Bu sınıfa ekleme, silme, arama öğeleri ve diğerleri gibi çift bağlantılı listelerle çalışmak için temel yöntemler sağlanmaktadır.

Deklarasyon

```
template<typename T>
class CLinkedList : public ICollection<T>
```

Başlık

```
#include <Generic\LinkedList.mqh>
```

Inheritance Hiyerarşisi

ICollection

CLinkedList

Sınıf Metodları

Metot	Tanım
Add	Bağlı bir listeye bir öğe ekler
AddAfter	Bağlantılı listede belirtilen düğümden sonra bir öğe ekler
AddBefore	Bağlantılı listede belirtilen düğümden önce bir öğe ekler
AddFirst	Bağlı listenin başına bir öğe ekler
AddLast	Bağlı listenin sonuna bir öğe ekler
Count	Bağlı listedeki öğelerin sayısını döndürür
Head	Bağlı listenin ilk düğümüne bir işaretçi döndürür
First	Bağlı listenin ilk düğümüne bir işaretçi döndürür
Last	Bağlı listenin son düğümüne bir işaretçi döndürür
Contains	Bağlı listede belirtilen değere sahip bir öğe olup olmadığını belirler.
CopyTo	Bağlantılı listenin tüm öğelerini, belirtilen dizinde başlayarak belirtilen diziye kopyalar
Clear	Bağlı listeden tüm öğeleri kaldırır
Remove	Belirtilen öğenin ilk bulunduğu öğeyi bağlantılı listeden kaldırır
RemoveFirst	Bağlı listenin ilk öğesini kaldırır.

Metot	Tanım
RemoveLast	Bağlı listenin son öğesini kaldırır
Find	Bağlantılı listede belirtilen değerin ilk bulunduğu yer için arama yapar
FindLast	Bağlı listede belirtilen değerin son ortaya çıkışını arar

Add

Bağlantılı listeye bir öğe ekler.

```
bool Add(
   T  value     // öğenin değeri
   );
```

Parametreler

value

[in] Eklenecek öğenin değeri

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

AddAfter

Bağlantılı listedeki belirli bir düğümünden sonra bir öğe ekler.

Bir öğeyi değere ekleyen versiyon.

```
CLinkedListNode<T>* AddAfter(
   CLinkedListNode<T>*   node,         // öğenin eklenmesi gereken düğüm
   T                     value         // eklemek için öğe
   );
```

Dönen Değer

Eklenen düğüme bir işaretçi döndürür.

Bir öğeyi değer olarak biçimlendirilmiş bir düğüm olarak ekleyen sürüm.

```
bool AddAfter(
   CLinkedListNode<T>*   node,         // öğenin eklenmesi gereken düğüm
   CLinkedListNode<T>*   new_node      // eklenecek düğüm
   );
```

Parametreler

node

[in] Yeni öğeni ekleneceği ilişkili listenin düğümü.

value

[in] Eklenecek öğe.

new_node

[in] Eklenecek düğüm.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

AddBefore

Bağlantılı listede belirtilen düğümden önce bir öğe ekler.

Bir öğeyi değere ekleyen versiyon.

```
CLinkedListNode<T>* AddBefore(
   CLinkedListNode<T>*  node,         // -den önce öğenin eklenmesi gereken düğüm
   T                    value         // eklemek için öğe
   );
```

Dönen Değer

Eklenen düğüme bir işaretçi döndürür.

Bir öğeyi değer olarak biçimlendirilmiş bir düğüm olarak ekleyen sürüm.

```
bool AddBefore(
   CLinkedListNode<T>*  node,         // -den önce öğenin eklenmesi gereken düğüm
   CLinkedListNode<T>*  new_node      // eklenecek düğüm
   );
```

Parametreler

node

[in] Yeni öğenin eklenmesinden önce ilişkili listenin düğümü.

value

[in] Eklenecek öğe.

**new_node*

[in] Eklenecek düğüm.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

AddFirst

Bağlı listenin başlangıcına bir öğe ekler.

Bir öğeyi değere ekleyen versiyon.

```
CLinkedListNode<T>* AddFirst(
   T    value                     // eklemek için bir öğe
   );
```

Dönen Değer

Eklenen düğüme bir işaretçi döndürür.

Bir öğeyi değer olarak biçimlendirilmiş bir düğüm olarak ekleyen sürüm.

```
bool AddFirst(
   CLinkedListNode<T>*  node      // eklemek için düğüm
   );
```

Parametreler

value

[in] Eklenecek öğe.

**node*

[in] Eklenecek düğüm.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

AddLast

Bağlı listenin sonuna bir öğe ekler

Bir öğeyi değere ekleyen versiyon.

```
CLinkedListNode<T>* AddLast(
   T  value                        // eklemek için öğe
   );
```

Dönen Değer

Eklenen düğüme bir işaretçi döndürür.

Bir öğeyi değer olarak biçimlendirilmiş bir düğüm olarak ekleyen sürüm.

```
bool AddLast(
   CLinkedListNode<T>*  node      // eklemek için düğüm
   );
```

Parametreler

value

[in] Eklenecek öğe.

**node*

[in] Eklenecek düğüm.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Count

İlişkili listedeki öğelerin sayısını döndürür.

```
int Count();
```

Dönen Değer

Öğelerin sayısını döndürür.

Head

Bağlı listenin ilk düğümüne bir işaretçi döndürür.

```
CLinkedListNode<T>* Head();
```

Dönen Değer

İlk düğüme bir işaretçi döndürür.

First

Bağlı listenin ilk düğümüne bir işaretçi döndürür.

```
CLinkedListNode<T>* First();
```

Dönen Değer

İlk düğüme bir işaretçi döndürür.

Last

Bağlı listenin son düğümüne bir işaretçi döndürür.

```
CLinkedListNode<T>* Last();
```

Dönen Değer

Son düğüme bir işaretçi döndürür.

Contains

Bağlı listede belirtilen değere sahip bir öğe olup olmadığını belirler.

```
bool Contains(
   T  item     // arama değeri
   );
```

Parametreler

item

[in] Aranan değer.

Dönen Değer

Bağlı listede belirtilen değere sahip bir öğe bulunursa true, aksi halde false döndürür.

CopyTo

Bağlantılı listenin tüm öğelerini, belirtilen indekste başlayarak belirtilen diziye kopyalar.

```
int CopyTo(
   T&          dst_array[],      // yazı için bir dizi
   const int   dst_start=0       // yazmak için başlangıç indeksi
   );
```

Parametreler

&dst_array[]

[out] İlişkili listenin öğelerinin yazılacağı bir dizi.

dst_start=0

[in] Kopyalamanın başladığı dizideki bir indeks.

Dönen Değer

Kopyalanan öğelerin sayısını döndürür.

Clear

Bir koleksiyonun tüm öğelerini kaldırır.

```
void Clear();
```

Remove

Belirtilen öğenin ilk bulunduğu varlığı ilişkili listeden kaldırır.

Bir öğeyi değerine göre kaldıran versiyon.

```
bool Remove(
   T  item                        // öğe değeri
   );
```

Bir düğümü işaret ederek bir öğeyi kaldıran versiyon.

```
bool Remove(
   CLinkedListNode<T>*  node      // öğe düğümü
   );
```

Parametreler

item

[in] Silinecek öğenin değeri.

**node*

[in] Silinecek öğenin düğümü.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

RemoveFirst

Bağlı listenin ilk öğesini kaldırır.

```
bool RemoveFirst();
```

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

RemoveLast

İlişkili listenin son öğesini kaldırır.

```
bool RemoveLast();
```

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Find

İlişkili listedeki belirtilen değerin ilk bulunduğu yeri arar.

```
CLinkedListNode<T>* Find(
   T  value     // arama değeri
   );
```

Parametreler

value

[in] Aranan değer.

Dönen Değer

Başarı durumunda arama değerini içeren ilk bulunmuş düğüme bir işaretçi döndürür, aksi halde NULL verir.

FindLast

İlişkili listede belirtilen değerin son ortaya çıkışını arar.

```
CLinkedListNode<T>* FindLast(
   T  value     // arama değeri
   );
```

Parametreler

value

[in] Aranan değer.

Dönen Değer

Başarılı olması halinde arama değerini içeren en son bulunan düğüme bir işaretçi döndürür, aksi halde NULL verir.

CQueue<T>

CQueue<T>, ICollection<T> arabirimini uygulayan jenerik bir sınıftır.

Tanım

The CQueue<T> sınıfı, FIFO (ilk giren ilk çıkar) ilkesinde çalışan bir sıraya göre düzenlenmiş T türü datanın bir dinamik koleksiyonudur.

Deklarasyon

```
template<typename T>
class CQueue : public ICollection<T>
```

Başlık

```
#include <Generic\Queue.mqh>
```

Inheritance Hiyerarşisi

ICollection

CQueue

Sınıf Metodları

Metot	Tanım
Add	Kuyruğa bir öğe ekler
Enqueue	Kuyruğa bir öğe ekler
Count	Kuyruktaki öğelerin sayısını döndürür
Contains	Kuyruğun belirtilen bir öğeye sahip olup olmadığını belirler
TrimExcess	Bir kuyruğun kapasitesini gerçek öğe sayısına ayarlar ve böylece kullanılmayan belleği boşa çıkarır
CopyTo	Belirtilen dizinden başlayarak belirtilen diziye bir kuyruğun tüm öğelerini kopyalar
Clear	Kuyruktan tüm öğeleri kaldırır
Remove	Kuyruktan belirli bir öğenin ilk var olduğu durumu kaldırır
Dequeue	Başlangıç öğesini döndürür ve onu kuyruktan kaldırır
Peek	Başlangıç öğesini, onu kuyruktan kaldırmadan döndürür

Add

Kuyruğa bir öğe ekler.

```
bool Add(
   T  value     // öğenin değeri
   );
```

Parametreler

value

[in] Eklenecek öğenin değeri

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Enqueue

Kuyruğa bir öğe ekler.

```
bool Enqueue(
   T  value     // eklemek için öğe
   );
```

Parametreler

value

[in] Eklenecek öğe.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Count

Kuyruktaki öğelerin sayısını döndürür.

```
int Count();
```

Dönen Değer

Öğelerin sayısını döndürür.

Contains

Bir kuyruğun belirli bir değerde bir öğe içerip içermediğini belirler.

```
bool Contains(
   T  item     // arama değeri
   );
```

Parametreler

item

[in] Aranan değer.

Dönen Değer

Belirtilen değere sahip bir öğe kuyrukta bulunuyor ise true, aksi halde false döndürür.

TrimExcess

Bir kuyruğun kapasitesini gerçek öğe sayısına ayarlar ve böylece kullanılmayan belleği boşa çıkarır.

```
void TrimExcess();
```

CopyTo

Belirtilen dizinden başlayarak belirtilen diziye bir kuyruğun tüm öğelerini kopyalar.

```
int CopyTo(
   T&          dst_array[],     // yazı için bir dizi
   const int   dst_start=0      // yazmak için başlangıç indeksi
   );
```

Parametreler

&dst_array[]

[out] Kuyruğun öğelerinin yazılacağı bir dizi.

dst_start=0

[in] Kopyalamanın başladığı dizideki bir indeks.

Dönen Değer

Kopyalanan öğelerin sayısını döndürür.

Clear

Bir kuyruktan tüm öğeleri kaldırır.

```
void Clear();
```

Remove

Kuyruktan belirli bir öğenin ilk varlığını kaldırır.

```
bool Remove(
   T  item     // öğe değeri
   );
```

Parametreler

item

[in] Silinecek öğenin değeri.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Dequeue

İlk öğeyi döndürür ve onu kuyruktan siler.

```
T Dequeue();
```

Dönen Değer

Başlangıç öğesini döndürür.

Peek

İlk öğeyi onu kuyruktan kaldırmadan döndürür.

```
T Peek();
```

Dönen Değer

Başlangıç öğesini döndürür.

CRedBlackTree<T>

CRedBlackTree<T>, ICollection<T> arabirimini uygulayan jenerik bir sınıftır.

Tanım

The CRedBlackTree<T> sınıfı T türü dataları depolayan bir dinamik red-black tree'nin bir uygulamasıdır. Sınıf, red-black treeler ile çalışmak için eklemek, silmek, maksimum ve minimum değeri aramak ve daha fazlası gibi temel metodları destekler.

Deklarasyon

```
template<typename T>
class CRedBlackTree : public ICollection<T>
```

Başlık

```
#include <Generic\RedBlackTree.mqh>
```

Inheritance Hiyerarşisi

ICollection

CRedBlackTree

Sınıf Metodları

Metot	Tanım
Add	Red-black tree'ye bir öğe ekler
Root	Red-black tree'nin kök dizinine bir işaretçi döndürür
Count	Red-black tree'deki öğelerin sayısını döndürür
Contains	Red-black tree'nin belirtilen bir değerdeki öğeyi içerip içermediğini belirler
Comparer	Red-black tree düzenlemek için kullanılan IComparer<T> arabirimine bir işaretçi döndürür
TryGetMin	Red-black tree'nin minimum öğesini alır
TryGetMax	Red-black tree'nin maksimum öğesini alır
CopyTo	Belirtilen diziden başlayarak red-black tree'nin tüm öğlerini belirtilen diziye kopyalar
Clear	Red-black treeden öğeleri kaldırır
Remove	Red-black treeden belirtilen öğenin varlığını kaldırır
RemoveMin	Bir red-black treedeki minimum değerdeki öğeyi kaldırır
RemoveMax	Bir red-black treedeki maksimum değerdeki öğeyi kaldırır
Find	Bir red-black treedeki belirli bir değerin varlığını arar

Metot	Tanım
FindMax	Bir red-black treedeki maksimum değerli öğeyi arar
FindMin	Bir red-black treedeki minimum değerli öğeyi arar

Add

Red-black treeye bir öğe ekler.

```
bool Add(
   T  value     // eklemek için öğe
   );
```

Parametreler

value

[in] Eklenecek öğe.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Count

Red-black ağacının öğe sayısını döndürür.

```
int Count();
```

Dönen Değer

Öğelerin sayısını döndürür.

Root

Red-black ağacının kök dizinine bir işaretçi döndürür.

```
CRedBlackTreeNode<T>* Root();
```

Dönen Değer

Kök dizine bir işaretçi döndürür.

Contains

Red-black ağacının belirtilen bir değerdeki öğeyi içerip içermediğini belirler.

```
bool Contains(
   T  item     // arama değeri
   );
```

Parametreler

item

[in] Aranan değer.

Dönen Değer

True döndürür eğer belirli bir değerdeki bir öğered-black ağacında bulunuyorsa, aksi halde false döndürür.

Comparer

Bir red-black ağacını düzenlemek için kullanılan IComparer<T> arabirimine bir işaretçi döndürür.

```
IComparer<T>* Comparer() const;
```

Dönen Değer

IComparer<T> arabirimine bir işaretçi döndürür.

TryGetMin

Red-black ağacının minimum öğesini alır.

```
bool TryGetMin(
   T&  min     // değer yazmak için bir değişken
   );
```

Parametreler

&min

[out] Minimum değerin yazılacağı değişken.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

TryGetMax

Red-black ağacının maksimum öğesini alır.

```
bool TryGetMax(
   T&  max     // değer yazmak için bir değişken
   );
```

Parametreler

&max

[out] Maksium değerin yazılacağı değişken.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

CopyTo

Belirtilen indexten başlayarak, red-black bir ağacının tüm öğelerini belirtilen diziye kopyalar.

```
int CopyTo(
   T&          dst_array[],      // yazmak için bir dizi
   const int   dst_start=0       // yazmak için başlangıç indeksi
   );
```

Parametreler

&dst_array[]

[out] red-black ağacının elemanlarının yazılacağı bir dizi.

dst_start=0

[in] Kopyalamanın başladığı dizideki bir indeks.

Dönen Değer

Kopyalanan öğelerin sayısını döndürür.

Clear

Bir red-black ağacından tüm öğeleri kaldırır.

```
void Clear();
```

Remove

Red-black ağacından belirli bir öğenin varlığını kaldırır.

Belirli bir değerdeki öğeyi kaldıran versiyon.

```
bool Remove(
   T  value                          // öğe değeri
   );
```

Bir düğümü işaret ederek bir öğeyi kaldıran versiyon.

```
bool Remove(
   CRedBlackTreeNode<T>*  node      // öğe düğümü
   );
```

Parametreler

item

[in] Silinecek öğenin değeri.

**node*

[in] Silinecek öğenin düğümü.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

RemoveMin

Red-black ağacından minimum değerli öğeyi kaldırır.

```
bool RemoveMin();
```

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

RemoveMax

Red-black ağacından maksimum değerli öğeyi kaldırır.

```
bool RemoveMax();
```

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Find

Red-black ağacında belirtilen bir değerin varlığını araştırır.

```
CRedBlackTreeNode<T>* Find(
   T  value     // arama değeri
   );
```

Parametreler

value

[in] Aranan değer.

Dönen Değer

Başarı durumunda arama değerini içeren düğüme bir işaretçi döndürür, aksi halde NULL verir.

FindMin

Red-black ağacında minimum değeri olan bir öğe arar.

```
CRedBlackTreeNode<T>* FindMin();
```

Dönen Değer

Başarı durumunda minimum değeri içeren düğüme bir işaretçi döndürür, aksi halde NULL verir.

FindMax

Red-black ağacında maksimum değeri olan bir öğe arar.

```
CRedBlackTreeNode<T>* FindMax();
```

Dönen Değer

Başarı durumunda maksimum değeri içeren düğüme bir işaretçi döndürür, aksi halde NULL verir.

CSortedMap<TKey, TValue>

CSortedMap<TKey,TValue>, IMap<TKey,TValue> arayüzünü uygulayan bir jenerik sınıftır.

Tanım

The CSortedMap<TKey,TValue> sınıfı, verileri anahtara göre sıralanmış anahtar/değer çifti olarak depolanan ve anahtar tekliği gereksinimini dikkate alarak dinamik bir hash tablosunun bir uygulamasıdır. Bu sınıf, anahtar/değer çifti aramak ve silmek için anahtara bir değer girmek, diğer bir anahtar/değer çifti aramak ve bunlara erişmek gibi bir hash tablo ile çalışmak için temel yöntemler sağlar.

Deklarasyon

```
template<typename TKey, typename TValue>
class CSortedMap : public IMap<TKey, TValue>
```

Başlık

```
#include <Generic\SortedMap.mqh>
```

Inheritance Hiyerarşisi

ICollection
  IMap
    CSortedMap

Sınıf Metodları

Metot	Tanım
Add	Hash tabloya bir anahtar/değer çifti ekler
Count	Ayrılmış hash tablodaki öğe sayısını döndürür
Contains	Sıralanan hash tablonun belirtilen anahtar/değer tablosunu içerip içermediğini belirler
ContainsKey	Sıralanmış hash tablonun anahtar/değer tablosunun belirtilen anahtara sahip olup olmadığını belirler
ContainsValue	Sıralanmış hash tablonun anahtar/değer tablosunun belirtilen değere sahip olup olmadığını belirler
Comparer	Sıralanmış bir hash tablosunu düzenlemek için kullanılan IComparer<T> arayüzüne bir işaretçi döndürür
CopyTo	Tüm anahtar/değer çiftlerini, belirtilen indeksten başlayarak sıralanmış hash tablodan belirtilen dizilere kopyalar
Clear	Sıralı hash tablodaki tüm öğeleri kaldırır
Remove	Sıralı hash tablodan anahtar/değer çiftinin ilk varlığını kaldırır

Metot	Tanım
TryGetValue	Sıralı hash tablodan belirli bir anahtara sahip bir öğeyi alır
TrySetValue	Sıralı hash tablodan belirli bir anahtara sahip bir anahtar/değer çiftini değiştirir

Add

Hash tablosuna bir anahtar/değer çifti ekler.

Oluşturulan anahtar/değer çiftini ekleyen bir versiyon.

```
bool Add(
   CKeyValuePair<TKeyTValue>*  pair     // anahtar/değer çifti
   );
```

Belirtilen anahtar ve değer ile yeni bir anahtar/değer çifti ekleyen bir versiyon.

```
bool Add(
   TKey    key,                         // key
   TValue  value                        // değer
   );
```

Parametreler

**pair*

[in] anahtar/değer çifti.

key

[in] Anahtar.

value

[in] Değer.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Count

Sıralı hash tablodaki tüm öğelerin sayısını döndürür.

```
int Count();
```

Comparer

Sıralı karma tabloyu düzenlemek için kullanılan IComparer<T> arabirimine bir işaretçi döndürür.

```
IComparer<TKey>* Comparer() const;
```

Dönen Değer

IComparer<T> arabirimine bir işaretçi döndürür.

Contains

Sıralı hash tablosunun belirli bir anahtar değer tablosu içerip içermediğini belirler.

Oluşturulan anahtar/değer çifti ile çalışma versiyonu.

```
bool Contains(
   CKeyValuePair<TKeyTValue>*  item     // anahtar/değer çifti
   );
```

Ayrı olarak ayarlanmış bir anahtar ve değer biçiminde bir anahtar/değer çifti ile çalışma versiyonu.

```
bool Contains(
   TKey    key,                         // key
   TValue  value                        // değer
   );
```

Parametreler

**item*

[in] anahtar/değer çifti.

key

[in] Anahtar.

value

[in] Değer.

Dönen Değer

True döndürür, eğer sıralı hash tablosu belirli bir anahtar ve değere sahip bir anahtar/değer çifti içeriyorsa, aksi halde false döndürür.

ContainsKey

Sıralı hash tablosunun anahtar/değer tablosubelirli bir anahtar ileiçerip içermediğini belirler.

```
bool ContainsKey(
   TKey  key     // anahtar
   );
```

Parametreler

key

[in] Anahtar.

Dönen Değer

Sıralı hash tablosu, belirtilen bir anahtarla anahtar/değer çiftini içeriyorsa true, aksi halde false döndürür.

ContainsValue

Depo edilmiş hash tablosunun belirli bir değer ile anahtar/değer tablosu içerip içermediğini belirler.

```
bool ContainsValue(
   TValue  value     // değer
   );
```

Parametreler

value

[in] Değer.

Dönen Değer

Sıralı hash tablo, belirtilen değere sahip anahtar/değer çiftini içeriyorsa true, aksi halde false döndürür.

CopyTo

Copies all key/value pairs from the sorted Sıralı hash tablonun to the specified arrays, starting at the specified index.

Hash tabloyu anahtar/değer çifti dizisine kopyalar.

```
int CopyTo(
   CKeyValuePair<TKeyTValue>*&  dst_array[],     // anahtar/değer çiftleri yazmak için
   const int                    dst_start=0      // yazmak için başlangıç indeksi
   );
```

Anahtar ve değerler için dizileri ayırmak amacıyla bir hash tablosunu kopyalayan versiyon.

```
int CopyTo(
   TKey&       dst_keys[],                      // anahtarlar yazmak için bir dizi
   TValue&     dst_values[],                    // değerler yazmak için bir dizi
   const int   dst_start=0                      // yazmak için başlangıç indeksi
   );
```

Parametreler

&dst_array[]

[out] Hash tablosundan tüm çiftlerin yazılacağı bir dizi.

&dst_keys[]

[out] Hash tablosundan tüm anahtarların yazılacağı bir dizi.

&dst_values[]

[out] Hash tablosundan tüm değerlerin yazılacağı bir dizi.

dst_start=0

[in] Kopyalamanın başladığı dizideki bir indeks.

Dönen Değer

Kopyalanan anahtar/değer çifti sayısını döndürür.

Clear

Sıralı hash tablodan tüm öğeleri kaldırır.

```
void Clear();
```

Remove

Sıralı hash tablosundan anahtar/değer çiftinin ilk varlığını kaldırır.

Oluşturulan anahtar/değer çifti temel alınarak bir anahtar/değer çifti kaldıran versiyon.

```
bool Remove(
   CKeyValuePair<TKeyTValue>*  item     // anahtar/değer çifti
   );
```

Anahtara dayalı bir anahtar/değer çifti kaldıran versiyon.

```
bool Remove(
   TKey  key                           // anahtar
   );
```

Parametreler

**item*

[in] anahtar/değer çifti.

key

[in] Anahtar.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

TryGetValue

Sıralı hash tablosundan belirli bir anahtardaki bir öğeyi alır.

```
bool TryGetValue(
   TKey     key,       // anahtar
   TValue&  value      // değer yazmak için bir değişken
   );
```

Parametreler

key

[in] Anahtar.

&value

[out] anahtar/değer çiftinin belirtilen değeri yazılacak değişken.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

TrySetValue

Belirli bir anahtardaki sıralı hash tablosundan anahtar/değer çiftinin değerini değiştirir.

```
bool TrySetValue(
   TKey    key,       // anahtar
   TValue  value      // yeni değer
   );
```

Parametreler

key

[in] Anahtar.

value

[in] Belirtilen anahtar/değer çiftine atanacak yeni değer.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

CSortedSet<T>

CSortedSet<T>, ISet<T> arayüzünü uygulayan bir jenerik sınıftır.

Tanım

The CSortedSet<T> sınıfı, her değerin tekli olması gereken T değerinin sıralı dinamik veri kümesinin uygulanmasıdır. Bu sınıf, kümeler ve ilgili işlemlerle çalışmak için temel yöntemler sağlar; örneğin kümelerin birleşimi ve kesişimi, strict ve strict olmayan alt kümelerin tanımı ve diğerleri.

Deklarasyon

```
template<typename T>
class CSortedSet : public ISet<T>
```

Başlık

```
#include <Generic\SortedSet.mqh>
```

Inheritance Hiyerarşisi

ICollection
  ISet
    CSortedSet

Sınıf Metodları

Metot	Tanım
Add	Bir öğeyi sıralı bir sete ekler
Count	Sıralı bir setteki öğelerin sayısını döndürür
Contains	Sıralı bir listenin belirli bir değerde bir öğe içerip içermediğini belirler.
Comparer	Sıralı bir seti düzenlemek için kullanılan IComparer<T> arayüzüne bir işaretçi döndürür
TryGetMin	Sıralı setten minimum öğeyi alır
TryGetMax	Sıralı setten maksimum öğeyi alır
CopyTo	Sıralı setteki tüm öğeleri, belirtilen indekste başlayan belirtilen diziye kopyalar
Clear	Bir sıralı setten tüm öğeleri kaldırır
Remove	Belirli öğenin varlığını, sıralanmış bir setten kaldırır.
ExceptWith	Geçerli koleksiyon ile geçmiş bir koleksiyon (dizi) arasındaki farkı üretir.
IntersectWith	Geçerli koleksiyonun kesişme işlemini ve geçmiş koleksiyonu (dizi) üretir.

Metot	Tanım
SymmetricExceptWith	Geçerli koleksiyon ile geçmiş bir koleksiyon (dizi) arasındaki simetrik farkın işleyişini üretir.
UnionWith	Geçerli koleksiyonun birleşimini ve geçmiş bir koleksiyonu (dizi) üretir.
IsProperSubsetOf	Geçerli sıralı kümenin belirtilen koleksiyon veya dizinin doğru bir alt kümesi olup olmadığını belirler.
IsProperSupersetOf	Geçerli sıralı kümenin belirtilen koleksiyon veya dizinin doğru bir üst kümesi olup olmadığını belirler.
IsSubsetOf	Geçerli sıralı kümenin belirtilen koleksiyon veya dizinin alt kümesi olup olmadığını belirler.
IsSupersetOf	Geçerli sıralı kümenin belirtilen koleksiyon veya dizinin üst kümesi olup olmadığını belirler.
Overlaps	Geçerli sıralı kümenin belirtilen koleksiyon veya dizeyle çakışıp çakmadığını belirler
SetEquals	Geçerli sıralı kümenin belirtilen koleksiyon veya dizinin tüm öğelerini içerip içermediğini belirler
GetViewBetween	Geçerli sıralı kümeden, minimum ve maksimum değerlerle belirtilen bir altkümeyi alır
GetReverse	Geçerli sıralı kümenin bir kopyasını alır; burada tüm öğeler ters sırayla düzenlenir

Add

Sıralı bir sete bir öğeyi ekler.

```
bool Add(
   T  value     // öğenin değeri
   );
```

Parametreler

value

[in] Eklenecek öğenin değeri

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Count

Sıralı setteki öğelerin sayısını döndürür.

```
int Count();
```

Dönen Değer

Öğelerin sayısını döndürür.

Contains

Sıralı bir kümenin, belirtilen değere sahip bir öğe içerip içermediğini belirler

```
bool Contains(
   T  item     // arama değeri
   );
```

Parametreler

item

[in] Aranan değer.

Dönen Değer

Belirtilen değere sahip bir öğe kümede bulunursa true, aksi halde false döndürür.

Comparer

Sıralı bir küme düzenlemek için kullanılan IComparer<T> arayüzüne bir işaretçi döndürür.

```
IComparer<T>* Comparer() const;
```

Dönen Değer

IComparer<T> arabirimine bir işaretçi döndürür.

TryGetMin

Sıralı kümeden minimum öğeyi alır.

```
bool TryGetMin(
   T&  min     // değer yazmak için bir değişken
   );
```

Parametreler

&min

[out] Minimum değerin yazılacağı değişken.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

TryGetMax

Sıralı kümeden maksimum öğeyi alır.

```
bool TryGetMax(
   T&  max     // değer yazmak için bir değişken
   );
```

Parametreler

&max

[out] Maksium değerin yazılacağı değişken.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

CopyTo

Sıralı setteki tüm öğeleri, belirtilen indeksteki belirtilen diziye kopyalar.

```
int CopyTo(
   T&          dst_array[],     // yazı için bir dizi
   const int   dst_start=0      // yazmak için başlangıç indeksi
   );
```

Parametreler

&dst_array[]

[out] Kümenin elemanlarının yazılacağı bir dizi.

dst_start=0

[in] Kopyalamanın başladığı dizideki bir indeks.

Dönen Değer

Kopyalanan öğelerin sayısını döndürür.

Clear

Sıralı bir kümeden tüm öğeleri kaldırır.

```
void Clear();
```

Remove

Belirtilen öğenin varlığını, sıralanmış kümeden kaldırır.

```
bool Remove(
   T  item     // öğe değeri
   );
```

Parametreler

item

[in] Silinecek öğenin değeri.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

ExceptWith

Geçerli koleksiyon ile geçmiş bir koleksiyonun (dizi) arasındaki farkı üretir. Belirtilen koleksiyonda (dizi) bulunan tüm öğeleri geçerli koleksiyonundan (dizi) kaldırır.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
void ExceptWith(
   ICollection<T>*  collection     // koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
void ExceptWith(
   T&  array[]                     // dizi
   );
```

Parametreler

**collection*

[in] Geçerli sıralı setten hariç tutlacak bir koleksiyon.

&collection[]

[in] Geçerli sıralı setten hariç tutalacak bir dizi.

Not

Sonuç, geçerli koleksiyona (dizi) yazdırılır.

IntersectWith

Geçerli koleksiyonun kesişme işlemini ve geçmiş koleksiyonu (dizi) üretir. Geçerli koleksiyonu yalnızca belirtilen koleksiyonda (dizi) bulunan öğeleri içerecek şekilde değiştirir.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
void IntersectWith(
   ICollection<T>*  collection     // koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
void IntersectWith(
   T&  array[]                     // dizi
   );
```

Parametreler

**collection*

[in] Geçerli küme ile kesişecek bir koleksiyon.

&collection[]

[in] Geçerli küme ile kesişecek bir dizi.

Not

Sonuç, geçerli koleksiyona (dizi) yazdırılır.

SymmetricExceptWith

Geçerli koleksiyon ile geçmiş bir koleksiyon (dizi) arasındaki simetrik farkın operasyonunu üretir. Geçerli koleksiyonu, yalnızca kaynak nesnede veya belirtilen koleksiyonda (dizilim) bulunan öğeleri içerecek şekilde değiştirir, ancak her ikisini birden değiştirmez.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
void SymmetricExceptWith(
   ICollection<T>*  collection     // koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
void SymmetricExceptWith(
   T&  array[]                     // dizi
   );
```

Parametreler

**collection*

[in] Simetrik bir fark üretmek için bir koleksiyon.

&collection[]

[in] Simetrik bir fark üretmek için bir dizi.

Not

Sonuç, geçerli koleksiyona (dizi) yazdırılır.

UnionWith

Geçerli koleksiyonun ve geçmiş bir koleksiyonun (dizi) birleşimini üretir. Belirtilen koleksiyon (dizi) öğelerinden eksik olanı geçerli koleksiyona (dizi) ekler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
void UnionWith(
   ICollection<T>*  collection     // koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
void UnionWith(
   T&  array[]                     // dizi
   );
```

Parametreler

**collection*

[in] Mevcut kümenin birleşeceği bir koleksiyon.

&collection[]

[in] Geçerli kümenin birleşeceği bir dizi.

Not

Sonuç, geçerli koleksiyona (dizi) yazdırılır.

IsProperSubsetOf

Geçerli sıralı setin belirtilen koleksiyon veya dizinin doğru bir alt kümesi olup olmadığını belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool IsProperSubsetOf(
   ICollection<T>*  collection     // ilişkiyi belirlemek için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool IsProperSubsetOf(
   T&  array[]                     // ilişkiyi belirlemek için bir dizi
   );
```

Parametreler

**collection*

[in] İlişkiyi belirlemek için bir koleksiyon.

&collection[]

[in] İlişkiyi belirlemek için bir dizi.

Dönen Değer

Eğer geçerli sıralı set bir uygun alt küme ise true, aksi halde false döndürür.

IsProperSupersetOf

Geçerli sıralı kümenin belirtilen koleksiyon veya dizinin doğru bir üst kümesi olup olmadığını belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool IsProperSupersetOf(
   ICollection<T>*  collection     // ilişkiyi belirlemek için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool IsProperSupersetOf(
   T&  array[]                     // ilişkiyi belirlemek için bir dizi
   );
```

Parametreler

**collection*

[in] İlişkiyi belirlemek için bir koleksiyon.

&collection[]

[in] İlişkiyi belirlemek için bir dizi.

Dönen Değer

Returns true if the current sorted set is a proper superset, or false otherwise.

IsSubsetOf

Geçerli sıralı kümenin belirtilen koleksiyon veya dizinin alt kümesi olup olmadığını belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool IsSubsetOf(
   ICollection<T>*  collection     // ilişkiyi belirlemek için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool IsSubsetOf(
   T&  array[]                     // ilişkiyi belirlemek için bir dizi
   );
```

Parametreler

**collection*

[in] İlişkiyi belirlemek için bir koleksiyon.

&collection[]

[in] İlişkiyi belirlemek için bir dizi.

Dönen Değer

Geçerli sıralı küme bir alt küme ise true, aksi halde false döndürür.

IsSupersetOf

Geçerli sıralı kümenin belirtilen koleksiyon veya dizinin üst kümesi olup olmadığını belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool IsSupersetOf(
   ICollection<T>*  collection     // ilişkiyi belirlemek için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool IsSupersetOf(
   T&  array[]                     // ilişkiyi belirlemek için bir dizi
   );
```

Parametreler

**collection*

[in] İlişkiyi belirlemek için bir koleksiyon.

&collection[]

[in] İlişkiyi belirlemek için bir dizi.

Dönen Değer

Geçerli sıralı küme bir üst küme ise true, aksi halde false döndürür.

Overlaps

Geçerli sıralı kümenin belirtilen koleksiyon veya diziyle çakışıp çakmadığını belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool Overlaps(
   ICollection<T>*  collection     // karşılaştırmak için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool Overlaps(
   T&  array[]                     // karşılaştırmak için bir dizi
   );
```

Parametreler

**collection*

[in] Çakışmayı belirlemek için bir koleksiyon.

&collection[]

[in] Çakışmayı belirlemek için bir dizi.

Dönen Değer

Geçerli sıralı küme ve bir koleksiyon veya bir dizi çakışırsa true, aksi halde false döndürür.

SetEquals

Geçerli sıralı kümenin belirtilen koleksiyon veya dizinin tüm öğelerini içerip içermediğini belirler.

ICollection<T> arabirimini uygulayan koleksiyonla çalışmak için bir versiyon.

```
bool SetEquals(
   ICollection<T>*  collection     // karşılaştırmak için bir koleksiyon
   );
```

Bir dizi ile çalışmak için bir versiyon.

```
bool SetEquals(
   T&  array[]                     // karşılaştırmak için bir dizi
   );
```

Parametreler

**collection*

[in] Öğeleri karşılaştırmak için bir koleksiyon.

&collection[]

[in] Öğeleri karşılaştırmak için bir dizi.

Dönen Değer

Geçerli sıralı küme belirtilen koleksiyon veya dizinin tüm öğelerini içeriyorsa true, aksi halde false döndürür.

GetViewBetween

Geçerli sıralı setten, minimum ve maksimum değerlerle belirtilen bir altkümeyi alır.

```
bool GetViewBetween(
   T&  array[],          // yazmak için bir dizi
   T   lower_value,      // minimum değer
   T   upper_value       // maksimum değer
   );
```

Parametreler

&array[]

[out] Alt küme yazmak için bir dizi.

lower_value

[in] Aralığın minimum değeri.

upper_value

[in] Aralığın maksimum değeri.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

GetReverse

Geçerli sıralı kümenin bir kopyasını alır; burada tüm öğeler ters sırayla düzenlenir.

```
bool GetReverse(
   T&  array[]     // yazmak için bir dizi
   );
```

Parametreler

&array[]

[out] Yazmak için bir dizi.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

CStack<T>

CStack<T>, ICollection<T> arayüzünü uygulayan jenerik bir sınıftır.

Tanım

CStack <T> sınıfı, LIFO (son giren ilk çıkar) ilkesinde çalışan T türü verinin bir dinamik koleksiyonudur.

Deklarasyon

```
template<typename T>
class CStack : public ICollection<T>
```

Başlık

```
#include <Generic\Stack.mqh>
```

Inheritance Hiyerarşisi

ICollection

CStack

Sınıf Metodları

Metot	Tanım
Add	Bir öğeyi bir yığına ekler
Count	Bir yığındaki öğelerin sayısını döndürür
Contains	Bir yığının belirli bir değerdeki bir öğeyi içerip içermediğini belirler
TrimExcess	Bir yığının kapasitesini öğelerin gerçek sayısına ayarlar
CopyTo	Bir yığının tüm öğelerini, belirtilen indeksten başlayarak belirtilen diziye kopyalar
Clear	Bir yığından tüm öğeleri kaldırır
Remove	Bir yığından belirli öğenin ilk varlığını kaldırır
Push	Bir öğeyi bir yığına ekler
Peek	Baş öğeyi bir yığından çıkarmadan döndürür
Pop	Baş öğeyi döndürür ve onu yığından kaldırır

Add

Yığına bir öğeyi ekler.

```
bool Add(
   T  value     // öğenin değeri
   );
```

Parametreler

value

[in] Eklenecek öğenin değeri

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Count

Bir yığındaki öğelerin sayısını döndürür.

```
int Count();
```

Dönen Değer

Öğelerin sayısını döndürür.

Contains

Bir yığının belirli bir değerdeki bir öğeyi içerip içermediğini belirler.

```
bool Contains(
   T  item     // arama değeri
   );
```

Parametreler

item

[in] Aranan değer.

Dönen Değer

Eğer belirli bir değerdeki bir öğe bir yığında bulunursa true, aksi halde false döndürür.

TrimExcess

Bir yığının kapasitesini gerçek öğe sayısına ayarlar ve böylece kullanılmayan belleği boşa çıkarır.

```
void TrimExcess();
```

CopyTo

Bir yığının tüm elemanlarını, belirtilen indeksten başlayarak belirtilen diziye kopyalar.

```
int CopyTo(
   T&          dst_array[],      // yazmak için bir dizi
   const int   dst_start=0       // yazmak için başlangıç indeksi
   );
```

Parametreler

&dst_array[]

[out] Yığın öğelerinin yazılacağı bir dizi.

dst_start=0

[in] Kopyalamanın başladığı dizideki bir indeks.

Dönen Değer

Kopyalanan öğelerin sayısını döndürür.

Clear

Bir yığından tüm öğeleri kaldırır.

```
void Clear();
```

Remove

Bir yığından belirli bir öğenin ilk varlığını kaldırır.

```
bool Remove(
   T  item     // öğe değeri
   );
```

Parametreler

item

[in] Silinecek öğenin değeri.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Push

Yığına bir öğeyi ekler.

```
bool Push(
   T  value     // eklenecek öğe
   );
```

Parametreler

value

[in] Eklenecek öğe.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Peek

Bir yığından baş öğeyi kaldırmaksızın döndürür.

```
T Peek();
```

Dönen Değer

Baş öğeyi döndürür.

Pop

Bir yığından baş öğeyi döndürür ve onu döndürür.

```
T Pop();
```

Dönen Değer

Baş öğeyi döndürür.

ArrayBinarySearch

Öğeleri karşılaştırmak için IComparable <T> arayüzünü kullanarak artan sıralı bir boyutlu bir dizide belirtilen değeri arar.

```
template<typename T>
int ArrayBinarySearch(
   T&              array[],          // arama için bir dizi
   const int       start_index,      // başlangıç indeksi
   const int       count,            // arama aralığı
   T               value,            // arama değeri
   IComparer<T>*   comparer          // karşılaştırmak için arayüz
   );
```

Parametreler

&array[]

[out] Aranacak dizin.

value

[in] Aranan değer.

**comparer*

[in] Bir arayüz öğeleri karşılaştırmak için.

start_index

[in] Aramanın başladığı başlangıç dizisi.

count

[in] Arama aralığının uzunluğu

Dönen Değer

Bulunan öğrenin dizinini döndürür. Arama değeri bulunamazsa, değeri en yakın olan en küçük öğenin dizinini döndürür.

ArrayIndexOf

Tek boyutlu bir dizideki bir değerin ilk ortaya çıkışını arar.

```
template<typename T>
int ArrayIndexOf(
   T&          array[],          // arama için bir dizi
   T           value,            // arama değeri
   const int   start_index,      // başlangıç indeksi
   const int   count             // arama aralığı
   );
```

Parametreler

&array[]

[out] Aranacak dizin.

value

[in] Aranan değer.

start_index

[in] Aramanın başladığı başlangıç dizisi.

count

[in] Arama aralığının uzunluğu

Dönen Değer

İlk bulunan öğenin dizinini döndürür. Eğer değer bulunmazsa -1 değeri döner.

ArrayLastIndexOf

Tek boyutlu bir dizideki bir değerin son ortaya çıkışını arar.

```
template<typename T>
int ArrayLastIndexOf(
   T&          array[],          // arama için bir dizi
   T           value,            // arama değeri
   const int   start_index,      // başlangıç indeksi
   const int   count             // arama aralığı
   );
```

Parametreler

&array[]

[out] Aranacak dizin.

value

[in] Aranan değer.

start_index

[in] Aramanın başladığı başlangıç dizisi.

count

[in] Arama aralığının uzunluğu

Dönen Değer

En son bulunan öğenin dizinini döndürür. Eğer değer bulunmazsa -1 değeri döner.

ArrayReverse

Tek boyutlu bir dizideki öğelerin sırasını değiştirir.

```
template<typename T>
bool ArrayReverse(
   T&          array[],          // kaynak dizin
   const int   start_index,      // başlangıç indeksi
   const int   count             // öğelerin sayısı
   );
```

Parametreler

&array[]

[out] Kaynak dizin.

start_index

[in] Başlangıç indeksi.

count

[in] İşleme katılan dizi öğelerinin sayısı.

Dönen Değer

Başarılı olduğunda true, aksi halde false döndürür.

Compare

İki değeri karşılaştırır; bunlardan biri "diğerinden büyük, küçük veya eşittir".

İki bool değerini karşılaştırmak için bir versiyon.

```
int Compare(
   const bool  x,          // ilk değer
   const bool  y           // ikinci değer
   );
```

İki char değerini karşılaştırmak için bir versiyon.

```
int Compare(
   const char  x,          // ilk değer
   const char  y           // ikinci değer
   );
```

İki uchar değerini karşılaştırmak için bir versiyon.

```
int Compare(
   const uchar  x,         // ilk değer
   const uchar  y          // ikinci değer
   );
```

İki short değeri karşılaştırmak için bir versiyon.

```
int Compare(
   const short  x,         // ilk değer
   const short  y          // ikinci değer
   );
```

İki ushort değeri karşılaştırmak için bir versiyon.

```
int Compare(
   const ushort  x,        // ilk değer
   const ushort  y         // ikinci değer
   );
```

İki color değeri karşılaştırmak için bir versiyon.

```
int Compare(
   const color  x,         // ilk değer
   const color  y          // ikinci değer
   );
```

İki int değeri karşılaştırmak için bir versiyon.

```
int Compare(
   const int  x,           // ilk değer
   const int  y            // ikinci değer
   );
```

İki uint değeri karşılaştırmak için bir versiyon.

```
int Compare(
   const uint  x,          // ilk değer
   const uint  y           // ikinci değer
   );
```

İki datetime değeri karşılaştırmak için bir versiyon.

```
int Compare(
   const datetime  x,      // ilk değer
   const datetime  y       // ikinci değer
   );
```

İki long değeri karşılaştırmak için bir versiyon.

```
int Compare(
   const long  x,          // ilk değer
   const long  y           // ikinci değer
   );
```

İki ulong değeri karşılaştırmak için bir versiyon.

```
int Compare(
   const ulong  x,         // ilk değer
   const ulong  y          // ikinci değer
   );
```

İki float değeri karşılaştırmak için bir versiyon.

```
int Compare(
   const float  x,         // ilk değer
   const float  y          // ikinci değer
   );
```

İki double değeri karşılaştırmak için bir versiyon.

```
int Compare(
   const double  x,        // ilk değer
   const double  y         // ikinci değer
   );
```

İki string değeri karşılaştırmak için bir versiyon.

```
int Compare(
   const string  x,        // ilk değer
   const string  y         // ikinci değer
   );
```

Diğer türdeki iki değeri karşılaştırmak için bir versiyon.

```
template<typename T>
int Compare(
```

```
   T  x,                  // ilk değer
   T  y                   // ikinci değer
   );
```

Parametreler

x

[in] İlk değer

y

[in] İkinci değer

Dönen Değer

İki karşılaştırılan değerin oranını ifade eden bir sayı döndürür:

- Eğer sonuç sıfırdan küçükse, x küçüktür y den (x<y)
- Eğer sonuç sıfıra eşitse, x eşittir y ye (x=y)
- Eğer sonuç sıfırdan büyükse, x büyüktür y den (x>y)

Not

Eğer T türü, IComparable <T> arabirimini uygulayan bir nesne ise, nesneler Compare yöntemine göre karşılaştırılacaktır. Bütün diğer durumlarda sıfır döndürür.

Equals

Eşitlik için iki değeri karşılaştırır.

```
template<typename T>
bool Equals(
   T  x,      // ilk değer
   T  y       // ikinci değer
   );
```

Parametreler

x

[in] İlk değer

y

[in] İkinci değer

Dönen Değer

Nesneler eşitse true, aksi halde false döndürür.

Not

Eğer T türü IEqualityComparable<T> arayüzünü uygulayan bir nesne ise, sonrasında nesneler Equals karşılaştırma metoduna göre karşılaştırılacaktır. Eşitlik için standart karşılaştırma diğer bütün durumlarda kullanılır.

GetHashCode

Hash kodun değerini hesaplar.

Bool türü ile çalışmak için bir versiyon.

```
int GetHashCode(
   const bool   value          // değer
   );
```

Char türü ile çalışmak için bir versiyon.

```
int GetHashCode(
   const char   value          // değer
   );
```

uchar türü ile çalışmak için bir versiyon.

```
int GetHashCode(
   const uchar   value         // değer
   );
```

short türü ile çalışmak için bir versiyon.

```
int GetHashCode(
   const short   value         // değer
   );
```

ushort türü ile çalışmak için bir versiyon.

```
int GetHashCode(
   const ushort   value        // değer
   );
```

color türü ile çalışmak için bir versiyon.

```
int GetHashCode(
   const color   value         // değer
   );
```

int türü ile çalışmak için bir versiyon.

```
int GetHashCode(
   const int   value           // değer
   );
```

unit türü ile çalışmak için bir versiyon.

```
int GetHashCode(
   const uint   value          // değer
   );
```

datetime türü ile çalışmak için bir versiyon.

```
int GetHashCode(
```

```
   const datetime   value      // değer
   );
```

long türü ile çalışmak için bir versiyon.

```
int GetHashCode(
   const long   value          // değer
   );
```

ulong türü ile çalışmak için bir versiyon.

```
int GetHashCode(
   const ulong   value         // değer
   );
```

float türü ile çalışmak için bir versiyon.

```
int GetHashCode(
   const float   value         // değer
   );
```

double türü ile çalışmak için bir versiyon.

```
int GetHashCode(
   const double   value        // değer
   );
```

string türü ile çalışmak için bir versiyon.

```
int GetHashCode(
   const string   value        // değer
   );
```

Diğer türlerle çalışmak için bir versiyon.

```
template<typename T>
int GetHashCode(
   T   value                   // değer
   );
```

Parametreler

value

[in] Hash kodunu almak istediğiniz değer.

Dönen Değer

Hash kodu döndürür.

Not

Eğer T türü IEqualityComparable<T> arayüzünü uygulayan bir nesne ise, sonrasında hash kodu, onun HashCode metoduna dayanarak alınacaktır. Diğer bütün durumlarda hash kod In all other cases, hash kod value türünün değeri olarak hesaplanacaktır.

Dosya İşlemleri

Bu bölüm, dosya işlemi sınıfları ile çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için gereken açıklamaları içermektedir.

Dosya işlem sınıfları, dosya gidi/çıktı işlemlerini içeren uygulamalar geliştirirken zaman kazandıracaktır.

MQL5 Standart Kütüphanesinin dosya işlem sınıfları terminalin Include\Files çalışma klasöründe yer almaktadır.

Sınıf	Açıklama
CFile	temel dosya işlemleri sınıfı
CFileBin	İkili dosya işlemleri sınıfı
CFileTxt	Metin dosyası işlemleri sınıfı

CFile

CFile sınıfı; CFileBin ve CFileTxt sınıfları için bir temel sınıftır.

Açıklama

CFile sınıfı, soyundan gelen sınıflar için MQL5 API dosya ve klasör işlemlerine kolay erişim sağlar.

Bildirim

```
class CFile: public CObject
```

Başlık

```
#include <Files\File.mqh>
```

Kalıtım hiyerarşisi

CObject

CFile

İlk nesil

CFileBin, CFilePipe, CFileTxt

Sınıf Yöntemleri

Özellikler	
Handle	Dosya tanıtıcı değerini alır
Filename	Dosya ismini alır
Flags	Dosya bayraklarını alır
SetUnicode	FILE_UNICODE bayrağını ayarlar/temizler
SetCommon	FILE_COMMON bayrağını ayarlar/temizler
Dosyalar için genel yöntemler	
Open	Dosya açar
Close	Dosya kapar
Delete	Dosya siler
IsExist	Dosyanın varlığını denetler
Copy	Dosyayı kopyalar
Move	Dosyayı isimlendirir/taşır
Size	Dosya boyutunu alır
Tell	Mevcut dosya konumunu alır

Özellikler	
Seek	Mevcut dosya konumunu ayarlar
Flush	Dosyayı diskten temizler
IsEnding	Dosya sonunu denetler
IsLineEnding	Satır sonunu denetler
Klasörler için genel yöntemler	
FolderCreate	Klasör oluşturur
FolderDelete	Klasörü siler
FolderClean	Klasörü temizler
Arama yöntemleri	
FileFindFirst	Dosya araması başlatır
FileFindNext	Dosya aramasını devam ettirir
FileFindClose	Arama işlemini sonlandırır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Handle

Açılmış dosyanın tanıtıcı değerini alır.

```
int  Handle()
```

Dönüş değeri

Sınıf örneğine atanan açılmış dosyanın tanıtıcı değeri. Atanmış bir dosya yoksa -1 dönüşü yapar.

FileName

Açılmış dosyanın ismini alır.

```
string  FileName()
```

Dönüş değeri

Sınıf örneğine atanan açılmış dosyanın ismi. Atanmış bir dosya yoksa "" dönüşü yapar.

Flags

Açılmış dosyanın bayrağını alır alır.

```
int  Flags()
```

Dönüş değeri

Sınıf örneğine atanan açılmış dosyanın bayrağı.

SetUnicode

FILE_UNICODE bayrağını ayarlar/temizler.

```
void  SetUnicode(
   bool  unicode      // Yeni bayrak değeri
   )
```

Parametreler

unicode

[in] FILE_UNICODE bayrağının yeni değeri.

Not

Dizgi işlemlerinin sonucu FILE_UNICODE bayrağına bağlıdır. Değer 'false' ise, ANSI kodlaması (bir bitlik semboller) kullanılır. Bayrak ayarlanmışsa UNICODE kodlaması kullanılır (iki bitlik semboller). Dosya zaten açılmış durumdaysa bayrak değiştirilemez.

SetCommon

FILE_COMMON bayrağını ayarlar/temizler.

```
void  SetCommon(
   bool  common      // Yeni bayrak değeri
   )
```

Parametreler

common

[in] FILE_COMMON bayrağının yeni değeri.

Not

FILE_UNICODE bayrağı mevcut çalışma klasörünü gösterir. Değer 'false' ise, mevcut çalışma klasörü olarak yerel terminal klasörü kullanılır. Değer 'true' ise, mevcut çalışma klasörü olarak genel klasör kullanılır. Dosya zaten açılmış durumdaysa bayrak değiştirilemez.

Open

Belirtilen dosyayı açar ve açma işlemi başarılı ise dosyayı sınıf örneğine atar.

```
int  Open(
   const string  file_name,        // Dosya ismi
   int           flags,            // Bayraklar
   short         delimiter=9       // Ayraç
   )
```

Parametreler

file_name

[in] Açılacak dosyanın ismi.

flags

[in] Dosya açma bayrakları.

delimiter=9

[in] CSV dosya ayraçları.

Dönüş değeri

Açılan dosyanın tanıtıcı değeri.

Not

Çalışma klasörü, SetCommon() yöntemiyle tanımlanan FILE_COMMON bayrağına bağlıdır.

Close

Sınıf örneğine atanan dosyayı kapatır.

```
void  Close()
```

Delete

Sınıf örneğine atanan dosyayı siler.

```
void  Delete()
```

Delete

Belirtilen dosyayı siler.

```
void  Delete(
   const string  file_name      // Dosya ismi
   )
```

Parametreler

file_name

[in] Silinecek dosya ismi.

Not

Çalışma klasörü, SetCommon() yöntemiyle tanımlanmış olan FILE_COMMON bayrağına bağlıdır.

IsExist

Dosyanın varlığını denetler

```
bool  IsExist(
   const string  file_name      // Dosya ismi
   )
```

Parametreler

file_name

[in] Denetlenecek dosyanın ismi.

Dönüş değeri

Dosya mevcutsa 'true'.

Copy

Dosya kopyalar.

```
bool  Copy(
   const string  src_name,      // Kaynak dosya ismi
   int           src_flag,      // Bayrak
   const string  dst_name,      // Kopya dosyanın ismi
   int           dst_flags      // Bayraklar
   )
```

Parametreler

src_name

[in] Kopyalanacak dosya.

src_flag

[in] Kopyalanacak dosyanın bayrakları (sadece FILE_COMMON kullanılır).

dst_name

[in] Kopya dosyanın ismi.

dst_flags

[in] Kopya dosyanın bayrakları (sadece FILE_REWRITE ve FILE_COMMON kullanılır).

Dönüş değeri

Başarılı ise 'true', dosya kopyalanamamışsa 'false'.

Move

Renames/moves file.

```
bool  Move(
   const string  src_name,      // Kaynak dosya ismi
   int           src_flag,      // Bayrak
   const string  dst_name,      // Kopya dosyanın ismi
   int           dst_flags      // Bayraklar
   )
```

Parametreler

src_name

[in] Taşınacak dosyaın ismi.

src_flag

[in] Kopyalanacak dosyanın bayrakları (sadece FILE_COMMON kullanılır).

dst_name

[in] Kopya dosyanın ismi.

dst_flags

[in] Kopya dosyanın bayrakları (sadece FILE_REWRITE ve FILE_COMMON kullanılır).

Dönüş değeri

Başarılı ise 'true', dosya taşınamazsa 'false'.

Size

Bayt cinsinden dosya boyutunu alır.

```
ulong  Size()
```

Dönüş değeri

Bayt cinsinden dosya boyutu. Atanmış bir dosya yoksa ULONG_MAX dönüşü yapar.

Tell

Mevcut dosya konumunu alır.

```
ulong  Tell()
```

Dönüş değeri

Mevcut dosya konumu. Atanmış bir dosya yoksa ULONG_MAX dönüşü yapar.

Seek

Mevcut dosya konumunu ayarlar.

```
void  Seek(
   long                offset,      // konum
   ENUM_FILE_POSITION  origin       // kaynak
   )
```

Parametreler

offset

[in] Bayt cinsinden dosya konumu (negatif olabilir).

origin

[in] Konum kaynağı.

Dönüş değeri

Başarılı ise 'true', dosya konumu değişmediyse 'false'.

Flush

Dosyanın tüm girdi/çıktı tamponlarının verilerini diskten temizler.

```
void  Flush()
```

IsEnding

Dosya sonunu denetler. Dosya okuma işlemi sırasında kullanılır.

```
bool  IsEnding()
```

Dönüş değeri

Okuma veya bulma işleminin ardından, dosya sonu başarıldıysa 'true' dönüşü yapar.

IsLineEnding

Satır sonu için dosyayı denetler. Dosya okuma işlemi sırasında kullanılır.

```
bool  IsLineEnding()
```

Dönüş değeri

Txt veya csv dosyası okuma işleminin (CR-LF karakterleri) ardından, satır sonu başarıldıysa 'true' dönüşü yapar.

FolderCreate

Yeni klasör oluşturur.

```
bool  FolderCreate(
   const string  folder_name      // Klasör ismi
   )
```

Parametreler

folder_name

[in] Oluşturulacak klasörün ismi. FILE_COMMON bayrağıyla tanımlanan klasörün bulunduğu adresi içerir.

Dönüş değeri

Başarılı ise 'true', klasör oluşturulamadıysa 'false'.

Not

Çalışma klasörü, SetCommon() yöntemiyle tanımlanmış olan FILE_COMMON bayrağına bağlıdır.

FolderDelete

Belirtilen dosyayı siler.

```
bool  FolderDelete(
   const string  folder_name      // Klasör ismi
   )
```

Parametreler

folder_name

[in] Silinecek klasörün ismi. FILE_COMMON bayrağıyla tanımlanan klasörün bulunduğu adresi içerir.

Dönüş değeri

Başarılı ise 'true', klasör silinemediyse 'false'.

Not

Çalışma klasörü, SetCommon() yöntemiyle tanımlanmış olan FILE_COMMON bayrağına bağlıdır.

FolderClean

Belirtilen klasörü temizler.

```
bool  FolderClean(
   const string  folder_name      // Klasör ismi
   )
```

Parametreler

folder_name

[in] Silinecek klasörün ismi. FILE_COMMON bayrağıyla tanımlanan klasörün bulunduğu adresi içerir.

Dönüş değeri

Başarılı ise 'true', dosya silinemediyse 'false'.

Not

Çalışma klasörü, SetCommon() yöntemiyle tanımlanmış olan FILE_COMMON bayrağına bağlıdır.

FileFindFirst

Belirtilen filtre ile arama başlatır.

```
int  FileFindFirst(
   const string  filter,         // Arama filtresi
   string&       file_name       // Dizgi referansı
   )
```

Parametreler

filter

[in] Arama filtresi.

file_name

[out] Bulunana ilk dosya için dizgi referansı.

Dönüş değeri

Başarılı olması durumunda tanıtıcı değere dönüş yapar, bu değer FileFindNext kullanarak yapılacak yeni arama için veya - filtreye karşılık gelen hiçbir dosya yoksa - INVALID_HANDLE dönüşü için kullanılabilir.

Not

Çalışma klasörü, SetCommon() yöntemiyle tanımlanmış olan FILE_COMMON bayrağına bağlıdır.

FileFindNext

FileFindFirst() fonksiyonu ile başlatılan aramayı devam ettirir.

```
bool  FileFindNext(
   int      search_handle,     // Arama işleyicisi
   string&  file_name          // Bulunan bir sonraki dosya için dizgi referansı
   )
```

Parametreler

search_handle

[in] FileFindFirst() yöntemiyle alınan arama işleyicisi.

file_name

[in] Başarılı şekilde bulunan bir sonraki dosyanın ismi için dizgi referansı.

Dönüş değeri

Başarılı ise 'true', filtreye karşılık gelen hiçbir dosya yoksa 'false'.

FileFindClose

Arama işleyicisini kapatır.

```
void  FileFindClose(
   int  search_handle      // Arama işleyicisi
   )
```

Parametreler

search_handle

[in] FileFindFirst() yöntemiyle alınan arama işleyicisi.

CFileBin

CFileBin sınıfı, ikili dosyalara kolay erişim için tasarlanmıştır.

Açıklama

CFileBin sınıfı, ikili dosyalara erişim sağlar.

Bildirim

```
class CFileBin: public CFile
```

Başlık

```
#include <Files\FileBin.mqh>
```

Kalıtım hiyerarşisi

CObject
  CFile
    CFileBin

Sınıf Yöntemleri

Açma yöntemleri	
Open	İkili dosyayı açar
Yazma yöntemleri	
WriteChar	char veya uchar tipli değişkeni yazar
WriteShort	short veya ushort tipli değişkeni yazar
WriteInteger	int veya uint tipli değişkeni yazar
WriteLong	long veya ulong tipli değişkeni yazar
WriteFloat	float tipli değişkeni yazar
WriteDouble	double tipli değişkeni yazar
WriteString	string tipli değişkeni yazar
WriteCharArray	char veya uchar tipli bir dizi yazar
WriteShortArray	short or ushort tipli bir dizi yazar
WriteIntegerArray	int veya uint tipli bir dizi yazar
WriteLongArray	long veya ulong tipli bir dizi yazar
WriteFloatArray	float tipli bir dizi yazar
WriteDoubleArray	double tipli bir dizi yazar
WriteObject	CObject sınıfının soyundan gelen bir örneğin verilerini yazar

Açma yöntemleri	
Okuma yöntemleri	
ReadChar	char veya uchar tipli değişkeni okur
ReadShort	short veya ushort tipli değişkeni okur
ReadInteger	int veya uint tipli değişkeni okur
ReadLong	long veya ulong tipli değişkeni okur
ReadFloat	float tipli değişkeni okur
ReadDouble	double tipli değişkeni okur
ReadString	string tipli değişkeni okur
ReadCharArray	char veya uchar tipli bir diziyi okur
ReadShortArray	short veya ushort tipli bir diziyi okur
ReadIntegerArray	int veya uint tipli bir diziyi okur
ReadLongArray	long veya ulong tipli bir diziyi okur
ReadFloatArray	float tipli bir diziyi okur
ReadDoubleArray	double tipli bir diziyi okur
ReadObject	CObject sınıfının soyundan gelen bir örneğin verilerini okur

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler 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

Belirtilen ikili dosyayı açar ve açma işlemi başarılı ise dosyayı sınıf örneğine atar.

```
int  Open(
   const string  file_name,      // dosya ismi
   int           flags           // bayraklar
   )
```

Parametreler

file_name

[in] Açılacak dosyanın ismi.

flags

[in] Dosya açma bayrakları (FILE_BIN bayrağının ayarlanması zorunludur).

Dönüş değeri

Açılan dosyanın tanıtıcı değeri.

WriteChar

char veya uchar tipli bir değişkeni dosyaya yazar.

```
uint  WriteChar(
   char  value      // Değer
   )
```

Parametreler

value

[in] Yazılacak değişken.

Dönüş değeri

Yazılan baytların sayısı.

WriteShort

short veya ushort tipli bir değişkeni dosyaya yazar.

```
uint  WriteShort(
   short  value      // değer
   )
```

Parametreler

value

[in] Yazılacak değişken.

Dönüş değeri

Yazılan baytların sayısı.

WriteInteger

int veya uint tipli bir değişkeni dosyaya yazar.

```
uint  WriteInteger(
   int  value      // Değer
   )
```

Parametreler

value

[in] Yazılacak değişken.

Dönüş değeri

Yazılan baytların sayısı.

WriteLong

long veya ulong tipli bir değişkeni dosyaya yazar.

```
uint  WriteLong(
   long  value      // Değer
   )
```

Parametreler

value

[in] Yazılacak değişken.

Dönüş değeri

Yazılan baytların sayısı.

WriteFloat

float tipli bir değişkeni dosyaya yazar.

```
uint  WriteFloat(
   float  value      // Değer
   )
```

Parametreler

value

[in] Yazılacak değişken.

Dönüş değeri

Yazılan baytların sayısı.

WriteDouble

double tipli bir değişkeni dosyaya yazar.

```
uint  WriteDouble(
   double  value      // değer
   )
```

Parametreler

value

[in] Yazılacak değişken.

Dönüş değeri

Yazılan baytların sayısı.

WriteString

string tipli bir değişkeni dosyaya yazar.

```
uint  WriteString(
   const string  value      // Değer
   )
```

Parametreler

value

[in] Yazılacak dizgi

Dönüş değeri

Yazılan baytların sayısı.

WriteString

string tipli bir değişkeni dosyaya yazar.

```
uint  WriteString(
   const string  value,     // Değer
   int           size       // Boyut
   )
```

Parametreler

value

[in] Yazılacak dizgi

size

[in] Number of bytes to write.

Dönüş değeri

Yazılan baytların sayısı.

WriteCharArray

char veya uchar tipli değişkenlerden oluşan bir diziyi dosyaya yazar.

```
uint  WriteCharArray(
   char&  array[],            // Dizinin referansı
   int    start_item=0,       // Başlangıç elemanı
   int    items_count=-1      // Eleman sayısı
   )
```

Parametreler

array[]

[in] Yazılacak dizi.

start_item=0

[in] Yazmaya başlanacak ilk elemanın indisi.

items_count=-1

[in] Yazılacak elemanların sayısı (-1 : tüm dizi).

Dönüş değeri

Yazılan baytların sayısı.

WriteShortArray

short veya ushort tipli değişkenlerden oluşan bir diziyi dosyaya yazar.

```
uint  WriteShortArray(
   short&  array[],            // Yazılacak dizi
   int     start_item=0,       // Başlangıç elemanı
   int     items_count=-1      // Yazılacak elemanların sayısı
   )
```

Parametreler

array[]

[in] Yazılacak dizi.

start_item=0

[in] Yazmaya başlanacak ilk elemanın indisi.

items_count=-1

[in] Yazılacak elemanların sayısı (-1 : tüm dizi).

Dönüş değeri

Yazılan baytların sayısı.

WriteIntegerArray

int veya uint tipli değişkenlerden oluşan bir diziyi dosyaya yazar.

```
uint  WriteIntegerArray(
   int&  array[],            // Yazılacak dizi
   int   start_item=0,       // Başlangıç elemanı
   int   items_count=-1      // Yazılacak elemanların sayısı
   )
```

Parametreler

array[]

[in] Yazılacak dizi.

start_item=0

[in] Yazmaya başlanacak ilk elemanın indisi.

items_count=-1

[in] Yazılacak elemanların sayısı (-1 : tüm dizi).

Dönüş değeri

Yazılan baytların sayısı.

WriteLongArray

long veya ulong tipli değişkenlerden oluşan bir diziyi dosyaya yazar.

```
uint  WriteLongArray(
   long&  array[],            // Yazılacak dizi
   int    start_item=0,       // Başlangıç elemanı
   int    items_count=-1      // Yazılacak elemanların sayısı
   )
```

Parametreler

array[]

[in] Yazılacak dizi.

start_item=0

[in] Yazmaya başlanacak ilk elemanın indisi.

items_count=-1

[in] Yazılacak elemanların sayısı (-1 : tüm dizi).

Dönüş değeri

Yazılan baytların sayısı.

WriteFloatArray

float tipli değişkenlerden oluşan bir diziyi dosyaya yazar.

```
uint  WriteFloatArray(
   float&  array[],            // Yazılacak dizi
   int     start_item=0,       // Başlangıç elemanı
   int     items_count=-1      // Yazılacak elemanların sayısı
   )
```

Parametreler

array[]

[in] Yazılacak dizi.

start_item=0

[in] Yazmaya başlanacak ilk elemanın indisi.

items_count=-1

[in] Yazılacak elemanların sayısı (-1 : tüm dizi).

Dönüş değeri

Yazılan baytların sayısı.

WriteDoubleArray

double tipli değişkenlerden oluşan bir diziyi dosyaya yazar.

```
uint  WriteDoubleArray(
   double&  array[],             // Yazılacak dizi
   int      start_item=0,        // Başlangıç elemanı
   int      items_count=-1       // Yazılacak elemanların sayısı
   )
```

Parametreler

array[]

[in] Yazılacak dizi.

start_item=0

[in] Yazmaya başlanacak ilk elemanın indisi.

items_count=-1

[in] Yazılacak elemanların sayısı (-1 : tüm dizi).

Dönüş değeri

Yazılan baytların sayısı.

WriteObject

CObject sınıfının soyundan gelen bir örneğin verilerini dosyaya yazar.

```
bool  WriteObject(
   CObject*  object      // Nesnenin referansı
   )
```

Parametreler

object

[in] Okunacak olan - CObject sınıfının soyundan gelen - örneğin referansı.

Dönüş değeri

Başarılı ise 'true', veri yazılamadıysa 'false'.

ReadChar

char veya uchar tipli değişkeni okur.

```
bool  ReadChar(
   char&  value      // Hedef değişken
   )
```

Parametreler

value

[in] char tipli hedef değişkeni.

Dönüş değeri

Başarılı ise 'true', veri okunamadıysa 'false'.

ReadShort

Dosyadan short veya ushort tipli bir değişken okur.

```
bool  ReadShort(
   short&  value
   )
```

Parametreler

value

[in] short veya ushort tipli hedef değişken.

Dönüş değeri

Başarılı ise 'true', veri okunamadıysa 'false'.

ReadInteger

Dosyadan int veya uint tipli bir değişken okur.

```
bool  ReadInteger(
   int&  value      // hedef değişken
   )
```

Parametreler

value

[in] int veya uint tipli hedef değişken.

Dönüş değeri

Başarılı ise 'true', veri okunamadıysa 'false'.

ReadLong

Dosyadan, long veya ulong tipli bir değişken okur.

```
bool  ReadLong(
   long&  value
   )
```

Parametreler

value

[in] long veya ulong tipli hedef değişken.

Dönüş değeri

Başarılı ise 'true', veri okunamadıysa 'false'.

ReadFloat

Dosyadan, float tipli bir değişkeni okur.

```
bool  ReadFloat(
   float&  value      // Hedef değişken
   )
```

Parametreler

value

[in] float tipli hedef değişken.

Dönüş değeri

Başarılı ise 'true', veri okunamadıysa 'false'.

ReadDouble

Dosyadan, double tipli bir değişken okur.

```
bool  ReadDouble(
   double&  value
   )
```

Parametreler

value

[in] double tipli hedef değişken.

Dönüş değeri

Başarılı ise 'true', veri okunamadıysa 'false'.

ReadString

Dosyadan string tipli bir değişken okur.

```
bool  ReadString(
   string&  value      // Hedef dizgi
   )
```

Parametreler

value

[in] string tipli hedef değişken.

Dönüş değeri

Başarılı ise 'true', veri okunamadıysa 'false'.

ReadString

Dosyadan string tipli bir değişken okur.

```
bool  ReadString(
   string&  value
   )
```

Parametreler

value

[in] string tipli hedef değişken.

Dönüş değeri

Başarılı ise 'true', veri okunamadıysa 'false'.

ReadCharArray

char veya uchar tipli değişkenlerden oluşan bir diziyi dosyadan okur.

```
bool  ReadCharArray(
   char&  array[],             // Hedef dizi
   int    start_item=0,        // Başlangıç elemanı
   int    items_count=-1       // Okunacak eleman sayısı
   )
```

Parametreler

array[]

[in] char veya uchar tipli dizinin referansı.

start_item=0

[in] Okumaya başlanacak ilk eleman.

items_count=-1

[in] Okunacak elemanların sayısı (-1 : dosya sonuna kadar oku).

Dönüş değeri

Başarılı ise 'true', veri okunamadıysa 'false'.

ReadShortArray

Dosyadan, short veya ushort tipli bir dizi okur.

```
bool  ReadShortArray(
   short&  array[],            // Hedef dizi
   int     start_item=0,       // Başlangıç elemanı
   int     items_count=-1      // Okunacak eleman sayısı
   )
```

Parametreler

array[]

[in] short veya ushort tipli hedef dizinin referansı.

start_item=0

[in] Okumaya başlanacak ilk eleman.

items_count=-1

[in] Okunacak elemanların sayısı (-1 : dosya sonuna kadar oku).

Dönüş değeri

Başarılı ise 'true', veri okunamadıysa 'false'.

ReadIntegerArray

Dosyadan, int veya uint tipli bir dizi okur.

```
bool  ReadIntegerArray(
   int&  array[],             // Hedef dizi
   int   start_item=0,        // Başlangıç elemanı
   int   items_count=-1       // Okunacak eleman sayısı
   )
```

Parametreler

array[]

[in] int veya uint tipli hedef dizinin referansı.

start_item=0

[in] Okumaya başlanacak ilk eleman.

items_count=-1

[in] Okunacak elemanların sayısı (-1 : dosya sonuna kadar oku).

Dönüş değeri

Başarılı ise 'true', veri okunamadıysa 'false'.

ReadLongArray

Dosyadan, long veya ulong tipli bir dizi okur.

```
bool  ReadLongArray(
   long&  array[],             // Hedef dizi
   int    start_item=0,        // Başlangıç elemanı
   int    items_count=-1       // Okunacak eleman sayısı
   )
```

Parametreler

array[]

[in] long veya ulong tipli hedef dizinin referansı.

start_item=0

[in] Okumaya başlanacak ilk eleman.

items_count=-1

[in] Okunacak elemanların sayısı (-1 : dosya sonuna kadar oku).

Dönüş değeri

Başarılı ise 'true', veri okunamadıysa 'false'.

ReadFloatArray

Dosyadan, float tipli bir diziyi okur

```
bool  ReadFloatArray(
   float&  array[],            // Hedef dizi
   int     start_item=0,       // Başlangıç elemanı
   int     items_count=-1      // Okunacak eleman sayısı
   )
```

Parametreler

array[]

[in] float tipli hedef dizinin referansı.

start_item=0

[in] Okumaya başlanacak ilk eleman.

items_count=-1

[in] Okunacak elemanların sayısı (-1 : dosya sonuna kadar oku).

Dönüş değeri

Başarılı ise 'true', veri okunamadıysa 'false'.

ReadDoubleArray

Dosyadan, double tipli bir diziyi okur.

```
bool  ReadDoubleArray(
   double&  array[],             // Hedef dizi
   int      start_item=0,        // Başlangıç elemanı
   int      items_count=-1       // Okunacak eleman sayısı
   )
```

Parametreler

array[]

[in] double tipli hedef dizinin referansı.

start_item=0

[in] Okumaya başlanacak ilk eleman.

items_count=-1

[in] Okunacak elemanların sayısı (-1 : dosya sonuna kadar oku).

Dönüş değeri

Başarılı ise 'true', veri okunamadıysa 'false'.

ReadObject

CObject sınıfının soyundan gelen bir örneğin verilerini okur.

```
bool  ReadObject(
   CObject*  object      // Nesnenin referansı
   )
```

Parametreler

object

[in] Okunacak olan - CObject sınıfının soyundan gelen - örneğin referansı.

Dönüş değeri

Başarılı ise 'true', veri okunamadıysa 'false'.

CFileTxt

CFileTxt sınıfı metin dosyalarına kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CFileTxt, metin dosyalarına erişim sağlar.

Bildirim

```
class CFileTxt: public CFile
```

Başlık

```
#include <Files\FileTxt.mqh>
```

Kalıtım hiyerarşisi

CObject
  CFile
    CFileTxt

Sınıf Yöntemleri

Açma yöntemleri	
Open	Bir metin dosyası açar
Yazma yöntemleri	
WriteString	Dosyaya string tipli bir değişken yazar
Okuma yöntemleri	
ReadString	Dosyadan string tipli bir değişken okur

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler 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

Belirtilen metin dosyasını açar ve açma işlemi başarılı ise dosyayı sınıf örneğine atar.

```
int  Open(
   const string  file_name,      // dosya ismi
   int           flags           // bayraklar
   )
```

Parametreler

file_name

[in] Açılacak dosyanın ismi.

flags

[in] Dosya açma bayrakları (FILE_TXT bayrağının ayarlanması zorunludur).

Dönüş değeri

Açılan dosyanın tanıtıcı değeri.

WriteString

WDosyaya string tipli bir değişken yazar.

```
uint  WriteString(
   const string  value      // Yazılacak dizgi
   )
```

Parametreler

value

[in] Yazılacak dizgi

Dönüş değeri

Yazılan baytların sayısı.

ReadString

Dosyadan string tipli bir değişken okur.

```
string  ReadString()
```

Dönüş değeri

Okunmuş olan dizgi.

Dizgi işlemleri

Bu bölüm, dizgi işlemi sınıfları ile çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için gereken açıklamaları içermektedir.

Dizgi işlemleri sınıfları, metin işleme eylemleri içeren uygulamalar geliştirirken zaman kazandıracaktır.

MQL5 Standart Kütüphanesinin dizgi sınıfları terminalin çalışma dizininde Include\Strings klasöründe yer alır.

Sınıf	Açıklama
CString	Dizgi işlemleri için bir sınıf

CString

CString, string tipli değişkenlere kolay erişim için tasarlanmış bir sınıftır.

Açıklama

CString sınıfı, soyundan gelen tüm sınıflar için MQL5 API dizgi fonksiyonlarına kolay erişim sağlar.

Bildirim

```
class CString: public CObject
```

Başlık

```
#include <Strings\String.mqh>
```

Kalıtım hiyerarşisi

CObject

CString

Sınıf Yöntemleri

Veri erişim yöntemleri	
Str	Bir dizgi alır
Len	Dizginin uzunluğunu alır
Copy	Bir dizgiyi kopyalar
Doldurma yöntemleri	
Fill	Dizgiyi belirtilen karakterle doldurur
Assign	Bir dizgi atar
Append	Bir dizgi iliştirir
Insert	Bir dizgi (araya) ekler
Karşılaştırma yöntemleri	
Compare	dizgileri karşılaştırır
CompareNoCase	Büyük/küçük harfe duyarsız dizgi karşılaştırması yapar
Alt-dizgi yöntemleri	
Left	Dizginin sol tarafından belli sayıda karakter alır
Right	Dizginin sağ tarafından belli sayıda karakter alır
Mid	Bir dizgiden belirtilen sayıda karakter alır

Veri erişim yöntemleri	
Kırpma/silme yöntemleri	
Trim	Bir dizgiden belirtilen karakter kümesinin başındaki ve sonundaki olayları kaldırır
TrimLeft	Bir dizgiden belirtilen karakter kümesinin tüm önde gelen örneklerini kaldırır
TrimRight	Bir dizgiden belirtilen karakter kümesinin tüm sondaki örneklerini kaldırır
Clear	Bir dizgiyi temizler
Dönüştürme yöntemleri	
ToUpper	Dizgiyi büyük harflere dönüştür.
ToLower	Dizgiyi küçük harflere dönüştür.
Reverse	Dizgiyi ters çevirir
Arama yöntemleri	
Find	Alt-dizginin ilk eşleşmesini arar
FindRev	Alt-dizginin son eşleşmesini arar
Remove	Alt-dizgiyi dizgiden siler
Replace	Alt-dizgilerin yerini değiştirir

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type

Str

Bir dizgi alır.

```
string  Str() const;
```

Dönüş değeri

Bir dizginin kopyası.

Len

Dizginin uzunluğunu alır.

```
uint  Len() const;
```

Dönüş değeri

Dizginin uzunluğu.

Copy

Bir dizgiyi referansı ile kopyalar.

```
void  Copy(
   string&  copy      // Referans
   ) const;
```

Parametreler

copy

[in] Kopyalanacak dizginin referansı.

Copy

Dizgiyi CString sınıf örneğine kopyalar.

```
void  Copy(
   CString*  copy      // Nesne tanımlayıcısı
   ) const;
```

Parametreler

copy

[in] CString sınıf nesnesinin tanımlayıcısı.

Fill

Dizgiyi belirtilen karakterle doldurur.

```
bool  Fill(
   short  character      // Karakter
   )
```

Parametreler

character

[in] Doldurulacak karakter.

Dönüş değeri

Başarılı ise 'true', dizgi doldurulamazsa 'false'.

Assign

Bir dizgi atar.

```
void  Assign(
   const string  str      // Atanacak dizgi
   )
```

Parametreler

str

[in] Atanacak dizgi.

Assign

Dizgiyi CString sınıf örneğine atar.

```
void  Assign(
   CString*  str      // Nesne tanımlayıcısı
   )
```

Parametreler

str

[in] Atama yapılacak CString sınıf örneğinin tanıtıcısı.

Append

Bir dizgi iliştirir.

```
void  Append(
   const string  str      // iliştirilecek dizgi
   )
```

Parametreler

str

[in] İliştirilecek dizgi.

Append

Dizgiyi CString sınıf örneğine iliştirir.

```
void  Append(
   CString*  string      // Nesne tanımlayıcısı
   )
```

Parametreler

string

[in] İliştirilecek CString sınıf örneğinin tanıtıcısı.

Insert

Belirtilen konumdan bir dizgi ekler.

```
uint  Insert(
   uint           pos,      // Konumdan
   const string   str       // Eklenecek dizgi
   )
```

Parametreler

pos

[in] Ekleme konumu.

str

[in] Eklenecek dizgi.

Dönüş değeri

Sonuç dizgisi uzunluğu.

Insert

CString sınıf örneğinin belirtilen konumuna bir dizi ekler.

```
uint  Insert(
   uint       pos,      // Konum
   CString*   str       // Nesne tanımlayıcısı
   )
```

Parametreler

pos

[in] Ekleme konumu.

str

[in] Eklenecek CString sınıf nesnesinin tanımlayıcısı.

Dönüş değeri

Sonuç dizgisi uzunluğu.

Compare

Dizgiyi karşılaştırır.

```
int  Compare(
   const string  str      // Karşılaştırılacak dizi
   ) const;
```

Parametreler

str

[in] Karşılaştırılacak dizi.

Dönüş değeri

Dizgiler eşitse 0 dönüşü yapar, sınıf dizgisi karşılaştırılacak dizgiden küçükse -1, büyükse 1 dönüşü yapar.

Compare

Dizgiyi CString sınıf örneğinin bir dizgisiyle karşılaştırır.

```
int  Compare(
   CString*  str      // Nesne tanımlayıcısı
   ) const;
```

Parametreler

str

[in] Karşılaştırma yapılacak CString sınıf örneğinin tanımlayıcısı.

Dönüş değeri

Dizgiler eşitse 0 dönüşü yapar, sınıf dizgisi karşılaştırılacak dizgiden küçükse -1, büyükse 1 dönüşü yapar.

CompareNoCase

Büyük/küçük harfe duyarsız dizgi karşılaştırması yapar.

```
int  CompareNoCase(
   const string  str      // Karşılaştırılacak dizi
   ) const;
```

Parametreler

str

[in] Karşılaştırılacak dizi.

Dönüş değeri

Dizgiler eşitse 0 dönüşü yapar, sınıf dizgisi karşılaştırılacak dizgiden küçükse -1, büyükse 1 dönüşü yapar.

CompareNoCase

Dizgiyi CString sınıf örneğinin bir dizgisiyle (büyük/küçük harfe duyarsız olarak) karşılaştırır.

```
int  CompareNoCase(
   CString*  str      // Nesne tanımlayıcısı
   ) const;
```

Parametreler

str

[in] Karşılaştırma yapılacak CString sınıf örneğinin tanımlayıcısı.

Dönüş değeri

Dizgiler eşitse 0 dönüşü yapar, sınıf dizgisi karşılaştırılacak dizgiden küçükse -1, büyükse 1 dönüşü yapar.

Left

Dizginin sol tarafından belli sayıda karakter alır.

```
string  Left(
   uint  count      // Karakterlerin sayısı
   )
```

Parametreler

count

[in] Karakterlerin sayısı.

Dönüş değeri

Elde edilen alt-dizgi.

Right

Dizginin sağ tarafından belli sayıda karakter alır.

```
string  Right(
   uint  count      // Karakterlerin sayısı
   )
```

Parametreler

count

[in] Karakterlerin sayısı.

Dönüş değeri

Elde edilen alt-dizgi.

Mid

Bir dizgiden belirtilen sayıda karakter alır.

```
string  Mid(
   uint  pos,        // Konum
   uint  count       // Karakterlerin sayısı
   )
```

Parametreler

pos

[in] Dizgi konumu.

count

[in] Karakterlerin sayısı.

Dönüş değeri

Elde edilen alt-dizgi.

Trim

Belirtilen karakter kümesinin başındaki ve sonundaki olayları (ve ' ','\t','\r','\n') dizgiden kaldırır.

```
int  Trim(
   const string  targets      // Kaldırılacak karakterlerin kümesi
   )
```

Parametreler

targets

[in] Kaldırılacak karakterlerin kümesi.

Dönüş değeri

Kaldırılan karakterlerin sayısı.

Örnek:

```
//--- CString::Trim için bir örnek
#include <Strings\String.mqh>
//---
void OnStart()
  {
   CString str;
   //---
   str.Assign("   \t\tABCD\r\n");
   printf("Kaynak dizgi '%s'",str.Str());
   //---
   str.Trim("DA-DA-DA");
   printf("Sonuç dizgisi '%s'",str.Str());
  }
```

TrimLeft

Belirtilen karakter kümesinin başındaki tüm olayları (ve ' ','\t','\r','\n') dizgiden kaldırır.

```
int  TrimLeft(
   const string  targets      // Kaldırılacak karakterlerin kümesi
   )
```

Parametreler

targets

[in] Kaldırılacak karakterlerin kümesi.

Dönüş değeri

Kaldırılan karakterlerin sayısı.

TrimRight

Belirtilen karakter kümesinin sonundaki tüm olayları (ve ' ','\t','\r','\n') dizgiden kaldırır.

```
int  TrimRight(
   const string  targets      // Kaldırılacak karakterlerin kümesi
   )
```

Parametreler

targets

[in] Kaldırılacak karakterlerin kümesi.

Dönüş değeri

Kaldırılan karakterlerin sayısı.

Clear

Bir dizgiyi temizler.

```
bool  Clear()
```

Dönüş değeri

Başarılı ise 'true', dizgi temizlenemezse 'false'.

ToUpper

Dizgiyi büyük harflere dönüştür.

```
bool  ToUpper()
```

Dönüş değeri

Başarılı ise 'true', dizgi dönüştürülemezse 'false'.

ToLower

Dizgiyi küçük harflere dönüştür.

```
bool  ToLower()
```

Dönüş değeri

Başarılı ise 'true', dizgi dönüştürülemezse 'false'.

Reverse

Dizgiyi ters çevirir.

```
void  Reverse()
```

Find

Alt-dizginin ilk eşleşmesini arar.

```
int  Find(
   uint           start,          // Konum
   const string   substring       // Aranacak alt-dizgi
   ) const;
```

Parametreler

start

[in] Aramaya başlanacak karakterin dizgi içindeki indisi.

substring

[in] Aranacak alt dizgi.

Dönüş değeri

İstenen alt-dizgiyle uyuşan ilk karakterin indisi; alt-dizgi bulunamazsa -1.

FindRev

Alt-dizginin son eşleşmesini arar

```
int  FindRev(
   const string  substring      // Alt-dizgi
   ) const;
```

Parametreler

substring

[in] Aranacak alt-dizgi.

Dönüş değeri

İstenen alt-dizgiyle uyuşan son karakterin indisi; alt-dizgi bulunamazsa -1.

Remove

Bir alt-dizgiyi dizgiden siler.

```
uint  Remove(
   const string  substring      // Kaldırılacak alt-dizgi
   )
```

Parametreler

substring

[in] Kaldırılacak alt-dizgi.

Dönüş değeri

Kaldırılan alt-dizgi sayısı.

Replace

Alt-dizgilerin yerini değiştirir.

```
uint  Replace(
   const string  substring,     // değiştirilecek alt-dizgi
   const string  newstring      // Yeni alt-dizgi
   )
```

Parametreler

substring

[in] Değiştirilecek alt-dizgi.

newstring

[in] Yeni alt-dizgi.

Dönüş değeri

Değiştirilen alt-dizgi sayısı.

Grafik Nesneleri

Bu bölüm, grafiksel nesne sınıflarıyla çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için açıklamalar içermektedir.

Grafiksel nesne sınıflarının kullanımı, kullanıcıya özel programlar (betikler, uzmanlar) oluştururken zamandan kazanmanızı sağlar.

MQL5 Standart Kütüphanesi (grafik nesneleri açısından) terminalin çalışma dizininde Include\ChartObjects klasöründe yer almaktadır.

Sınıf/Grup	Açıklama
CChartObject	Grafik nesnelerinin temel sınıfı
Çizgiler	Grup sınıfları "Çizgiler"
Kanallar	Grup sınıfları "Kanallar"
Gann Araçları	Grup sınıfları "Gann"
Fibonacci Araçları	Grup sınıfları "Fibonacci"
Elliott Araçları	Grup sınıfları "Elliott"
Şekiller	Grup sınıfları "Şekiller"
Oklar	Grup sınıfları "Oklar"
Kontroller	Grup sınıfları "Kontroller"

CChartObject

CChartObject, Standard MQL5 kütüphanesinin çizelge tipi grafik nesneleri için temel bir sınıftır.

Açıklama

CChartObject sınıfı, kendisinden türetilen tüm sınıflar için MQL5 API fonksiyonlarına kolay erişim sağlar.

Bildirim

```
class CChartObject : public CObject
```

Başlık

```
#include <ChartObjects\ChartObject.mqh>
```

Kalıtım hiyerarşisi

CObject

CChartObject

İlk nesil

CChartObjectArrow, CChartObjectBitmap, CChartObjectBmpLabel, CChartObjectCycles, CChartObjectElliottWave3, CChartObjectEllipse, CChartObjectFiboArc, CChartObjectFiboFan, CChartObjectFiboTimes, CChartObjectHLine, CChartObjectRectangle, CChartObjectSubChart, CChartObjectText, CChartObjectTrend, CChartObjectTriangle, CChartObjectVLine

Sınıf Yöntemleri

Özellikler	
ChartId	Grafiği içeren çizelgenin tanımlayıcısını alır
Window	Grafik nesnesini içeren çizelge penceresinin numarasını alır
Name	Bir grafik nesnesinin ismini alır/ayarlar
NumPoints	Tutturma noktasının koordinatlarını alır
Atama	
Attach	Bir grafiksel nesneyi sınıf örneğine bağlar
SetPoint	Tutturma noktasını ayarlar
Silme	
Delete	Bir grafiksel nesneyi siler
Detach	Bir grafiksel nesneyi sınıf örneğinden ayırır
Kaydırma	
ShiftObject	Belirtilen grafiksel nesneyi kaydırır

Özellikler	
ShiftPoint	Belirtilen grafiksel nesnenin tutturma noktasını kaydırır
Nesne özellikleri	
Time	Belirtilen grafiksel nesnenin tutturma noktasının zaman koordinatını alır/ayarlar
Price	Belirtilen grafiksel nesnenin tutturma noktasının fiyat koordinatını alır/ayarlar
Color	Belirtilen grafiksel nesnenin rengini alır/ayarlar
Style	Belirtilen grafiksel nesnenin çizgi stilini alır/ayarlar
Width	Belirtilen grafiksel nesnenin genişliğini alır/ayarlar
BackGround	Belirtilen grafiksel nesnenin 'arka-plana çizim' bayrağını alır/ayarlar
Selected	Bir grafik nesnesinin "seçili" bayrağının değerini alır/ayarlar
Selectable	Grafiksel nesnenin "seçilebilir" bayrağını alır/ayarlar
Description	Belirtilen grafiksel nesnenin açıklama metnini alır/ayarlar
Tooltip	Belirtilen grafiksel nesnenin araç-ipucunu alır/ayarlar
Timeframes	Grafiksel nesnenin görünürlük bayraklarını (hangi periyotlarda görünür olduğu bilgisini) alır/ayarlar
Z_Order	Grafiksel nesnenin çizelge üzerindeki tıklama önceliği değerini alır/ayarlar
CreateTime	Belirtilen grafiksel nesnenin oluşturulma zamanını alır
Nesnenin seviye özellikleri	
LevelsCount	Nesne üzerinde tanımlı seviyelerin sayısını alır/ayarlar
LevelColor	Seviye çizgisinin rengini alır/ayarlar
LevelStyle	Seviye çizgisinin stilini alır/ayarlar
LevelWidth	Seviye çizgisinin genişliğini alır/ayarlar
LevelValue	Seviyenin değerini alır/ayarlar
LevelDescription	Seviyenin açıklama metnini alır/ayarlar
MQL5 API fonksiyonlarına erişim	
GetInteger	Nesne özelliğinin tamsayı tipli değerini alır
SetInteger	Nesne özelliğinin tamsayı tipli değerini ayarlar
GetDouble	Çift-duyarlı nesne özelliğinin değerini alır
SetDouble	Çift-duyarlı nesne özelliğinin değerini ayarlar

Özellikler	
GetString	Metin biçimli nesne özelliğinin değerini alır
SetString	Metin biçimli nesne özelliğinin değerini ayarlar
Giriş/Çıkış	
virtual Save	Sanal dosya giriş yöntemi
virtual Load	Sanal dosya okuma yöntemi
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

ChartId

Grafiksel nesneyi içeren çizelgenin tanımlayıcısını alır.

```
long  ChartId() const
```

Dönüş Değeri

Grafiksel nesneyi içeren çizelgenin tanımlayıcı. Nesne bulunamazsa -1 dönüşü yapar.

Örnek:

```
//--- CChartObject::ChartId için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- grafik nenesinin çizelge tanımlayıcısını al
   long chart_id=object.ChartId();
  }
```

Window

Grafik nesnesinin bulunduğu çizelge alt-penceresinin numarasını alır.

```
int  Window() const
```

Dönüş Değeri

Grafik nesnesinin bulunduğu çizelge alt-penceresinin numarası (0 - ana pencere). Nesne bulunamazsa -1 dönüşü yapar.

Örnek:

```
//--- CChartObject::Window için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- Nesnenin bulunduğu pencerenin numarasını al
   int window=object.Window();
  }
```

Name (Get Method)

Grafik nesnesinin ismini alır.

```
string  Name() const
```

Dönüş Değeri

Bir sınıf örneğine tutturulmuş grafiksel nesnenin ismi. Nesne bulunamaz ise NULL dönüşü yapar.

Name (Set Method)

Grafik nesnesinin ismini ayarlar.

```
bool  Name(
   string  name      // yeni isim
   )
```

Parametreler

name

[in] Grafik nesnesinin yeni ismi.

Dönüş Değeri

Başarılı ise 'true', nesne ismi değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::Name için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- grafik nesnesinin ismini al
   string object_name=object.Name();
   if(object_name!="MyChartObject")
     {
     //--- grafik nesnesinin ismini ayarla
     object.Name("MyChartObject");
     }
  }
```

NumPoints

Grafik nesnesinin tutturma noktalarının sayısını alır.

```
int  NumPoints() const
```

Dönüş Değeri

Bir sınıf örneğine bağlanmış grafiksel nesnenin tutturma noktalarının sayısı. Tutturulmuş bir nesne yoksa 0 dönüşü yapar.

Örnek:

```
//--- CChartObject::NumPoints için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- grafik nesnesinin tutturma noktalarının sayısını al
   int points=object.NumPoints();
  }
```

Attach

Bir grafiksel nesneyi bir sınıf örneğine tutturur.

```
bool  Attach(
   long     chart_id,      // Çizelge tanımlayıcısı
   string   name,          // Nesne ismi
   int      window,        // Çizelge penceresi
   int      points         // Noktaların sayısı
   )
```

Parametreler

chart_id

[out] Çizelge tanımlayıcısı.

name

[in] Grafiksel nesnenin ismi.

window

[in] Çizelge penceresinin numarası (0 - ana pencereyi temsil eder).

points

[in] Grafik nesnesinin tutturma noktalarının sayısı.

Dönüş Değeri

İşlem başarılı ise 'true', aksi durumda 'false'.

Örnek:

```
//--- CChartObject::Attach için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- grafik nesnesini tuttur
   if(!object.Attach(ChartID(),"MyObject",0,2))
     {
      printf("Nesne tutturma hatası");
      return;
     }
  }
```

SetPoint

Grafiksel nesnenin belirtilen tutturma noktası için yeni koordinatlar ayarlar.

```
bool  SetPoint(
   int       point,          // nokta numarası
   datetime  new_time,       // zaman koordinatı
   double    new_price       // fiyat koordinatı
   )
```

Parametreler

point

[in] Tutturma noktasının numarası.

new_time

[in] Belirtilen tutturma noktası için yeni zaman koordinatı değeri.

new_price

[in] Belirtilen tutturma noktası için yeni fiyat koordinatı değeri.

Dönüş Değeri

Başarılı ise 'true', koordinatlar değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::SetPoint için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   double       price;
   //---
   if(object.NumPoints()>0)
     {
      //--- grafik nesnesinin tutturma noktasını ayarla
      object.SetPoint(0,CurrTime(),price);
     }
  }
```

Delete

Bir grafiksel nesneyi tutturulduğu çizelgeden kaldırır.

```
bool  Delete()
```

Dönüş Değeri

Başarılı ise 'true', kaldırma başarısız olmuşsa 'false'.

Örnek:

```
//--- CChartObject::Delete için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- grafik nesnesini kaldır
   if(!object.Delete())
     {
      printf("Nesne silme hatası");
      return;
     }
  }
```

Detach

Graffik nesnesini tutturulduğu sınıf örneğinden ayırır.

```
void  Detach()
```

Dönüş Değeri

Yok.

Örnek:

```
//--- CChartObject::Detach örneği
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- grafik nesnesini ayır
   object.Detach();
  }
```

ShiftObject

Grafik nesnesini belirtilen ölçüde kaydırır (taşır).

```
bool  ShiftObject(
   datetime  d_time,      // zaman koordinatındaki artış
   double    d_price      // fiyat koordinatındaki artış
   )
```

Parametreler

d_time

[in] Nesnenin tutturma noktalarının zaman koordinatındaki artış miktarı.

d_price

[in] Nesnenin tutturma noktalarının fiyat koordinatındaki artış miktarı.

Dönüş Değeri

Başarılı ise 'true', nesne kaydırılamadıysa 'false'.

Örnek:

```
//--- CChartObject::ShiftObject için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   datetime     d_time;
   double       d_price;
   //--- grafik nesnesini kaydır
   object.ShiftObject(d_time,d_price);
  }
```

ShiftPoint

Grafik nesnesinin belirtilen tutturma noktasını kaydırır.

```
bool  ShiftPoint(
   int       point,       // nokta numarası
   datetime  d_time,      // zaman koordinatındaki artış
   double    d_price      // fiyat koordinatındaki artış
   )
```

Parametreler

point

[in] Tutturma noktasının numarası.

d_time

[in] Belirtilen tutturma noktasının zaman koordinatındaki artış miktarı.

d_price

[in] Belirtilen tutturma noktasının fiyat koordinatındaki artış miktarı.

Dönüş Değeri

Başarılı ise 'true', nokta kaydırılamadıysa 'false'.

Örnek:

```
//--- CChartObject::ShiftPoint için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   datetime     d_time;
   double       d_price;
   //---
   if(object.NumPoints()>0)
     {
      //--- grafik nesnesinin tutturma noktasını kaydır
      object.ShiftPoint(0,d_time,d_price);
     }
  }
```

Time (Get Method)

Grafik nesnesinin belirtilen tutturma noktasının zaman koordinatını alır.

```
datetime  Time(
   int  point      // nokta numarası
   ) const
```

Parametreler

point

[in] Tutturma noktasının numarası.

Dönüş Değeri

Bir sınıf örneğine tutturulmuş grafiksel nesnenin belirtilen tutturma noktasının zaman koordinatı. Tutturulmuş bir nesne yoksa veya nesne belirtilen noktaya sahip değilse 0 değerine dönüş yapar.

Time (Set Method)

Grafik nesnesinin belirtilen tutturma noktasının zaman koordinatını ayarlar.

```
bool  Time(
   int       point,        // nokta numarası
   datetime  new_time      // yeni zaman koordinatı
   )
```

Parametreler

point

[in] Tutturma noktasının numarası.

new_time

[in] Belirtilen tutturma noktası için yeni zaman koordinatı değeri.

Dönüş Değeri

Başarılı ise 'true', koordinat değeri değiştirilemediyse 'false'.

Örnek:

```
//--- CChartObject::Time için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.NumPoints();i++)
     {
      //--- grafik nesnesinin tutturma noktasının zaman koordinatını al
      datetime point_time=object.Time(i);
      if(point_time==0)
```

```
         {
          //--- grafik nesnesinin tutturma noktasının zaman koordinatını ayarla
          object.Time(i,TimeCurrent());
         }
      }
   }
```

Price (Get Method)

Grafik nesnesinin belirtilen tutturma noktasının fiyat koordinatını alır.

```
double  Price(
   int  point      // nokta numarası
   ) const
```

Parametreler

point

[in] Tutturma noktasının numarası.

Dönüş Değeri

Bir sınıf örneğine tutturulmuş grafiksel nesnenin belirtilen tutturma noktasının fiyat koordinatı. Tutturulmuş bir nesne yoksa veya nesne belirtilen noktaya sahip değilse EMPTY_VALUE değerine dönüş yapar.

Price (Set Method)

Grafik nesnesinin belirtilen tutturma noktasının fiyat koordinatını ayarlar.

```
bool  Price(
   int     point,          // Nokta numarası
   double  new_price       // Fiyat
   )
```

Parametreler

point

[in] Tutturma noktasının numarası.

new_price

[in] Belirtilen tutturma noktasının fiyat koordinatı için yeni değer.

Dönüş Değeri

Başarılı ise 'true', koordinat değeri değiştirilemediyse 'false'.

Örnek:

```
//--- CChartObject::Price için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   double       price;
   //---
   for(int i=0;i<object.NumPoints();i++)
     {
      //--- grafik nesnesinin tutturma noktasının fiyat koordinatını al
```

```
      double point_price=object.Price(i);
      if(point_price!=price)
        {
         //--- grafik nesnesinin tutturma noktasının fiyat koordinatını ayarla
         object.Price(i,price);
        }
     }
  }
```

Color (Get Method)

Grafik nesnenin çizgi rengini alır.

```
color  Color() const
```

Dönüş Değeri

Sınıf örneğine tutturulmuş grafik nesnesinin çizgi rengi. Tutturulmuş hiçbir nesne yoksa CLR_NONE değerine dönüş yapar.

Color (Set Method)

Grafik nesnenin çizgi rengini ayarlar.

```
bool  Color(
   color  new_color      // yeni renk
   )
```

Parametreler

new_color

[in] çizgi rengi için yeni değer.

Dönüş Değeri

Başarılı ise 'true', renk değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::Color için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- Grafik nesnesinin rengini al
   color object_color=object.Color();
   if(object_color!=clrRed)
     {
     //--- Grafik nesnesinin rengini ayarla
     object.Color(clrRed);
     }
  }
```

Style (Get Method)

Grafik nesnesinin çizgi stilini alır.

```
ENUM_LINE_STYLE  Style() const
```

Dönüş Değeri

Sınıf örneğine tutturulmuş grafik nesnesinin çizgi stili. Tutturulmuş bir nesne yoksa WRONG_VALUE dönüşü yapar.

Style (Set Method)

Grafik nesnesinin çizgi stilini ayarlar.

```
bool  Style(
   ENUM_LINE_STYLE  new_style      // yeni çizgi stili
   )
```

Parametreler

new_style

[in] Çizgi stilinin yeni değeri.

Dönüş Değeri

Başarılı ise 'true', stil değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::Style için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- grafik nesnesinin çizgi stilini al
   ENUM_LINE_STYLE style=object.Style();
   if(style!=STYLE_SOLID)
     {
      //--- grafik nesnesinin çizgi stilini ayarla
      object.Style(STYLE_SOLID);
     }
  }
```

Width (Get Method)

Grafik nesnenin çizgi kalınlığını alır.

```
int  Width() const
```

Dönüş Değeri

Bir sınıf örneğine tutturulmuş grafik nesnesinin çizgi kalınlığı. Tutturulmuş bir nesne yoksa -1 dönüşü yapar.

Width (Set Method)

grafik nesnesinin çizgi kalınlığını ayarlar.

```
bool  Width(
   int  new_width      // yeni kalınlık
   )
```

Parametreler

new_width

[in] Çizgi kalınlığı için yeni değer.

Dönüş Değeri

Başarılı ise 'true', kalınlık değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::Width için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- grafik nesnesinin çizgi kalınlığını al
   int width=object.Width();
   if(width!=1)
     {
      //--- grafik nesnesinin çizgi kalınlığını ayarla
      object.Width(1);
     }
  }
```

Background (Get Yöntemi)

Grafiksel nesnenin 'arka-plana çizim' bayrağını alır.

```
bool  Background() const
```

Dönüş Değeri

Bir sınıf örneğine bağlanmış olan grafik nesnesinin 'arka-plana çizilme' bayrağını alır. Nesne bağlanmamışsa 'false' dönüşü yapar.

Background (Set Yöntemi)

Grafik nesnesinin 'arka-plana çizim' bayrağını ayarlar.

```
bool  Background(
   bool  background      // Bayrak değeri
   )
```

Parametreler

background

[in] Grafiksel nesnenin arka-plana çizilme bayrağı için yeni değer.

Dönüş Değeri

Başarılı ise 'true', bayrak değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::Background için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- grafik nesnesinin arka-plan bayrağını al
   bool background_flag=object.Background();
   if(!background_flag)
     {
     //--- grafik nesnesinin arka-plan bayrağını ayarla
     object.Background(true);
     }
  }
```

Selected (Get Method)

Grafik nesnesinin seçili durumda olup olmadığını gösteren bayrağı alır Diğer bir deyişle nesne ya seçili durumdadır ya da değildir.

```
bool  Selected() const
```

Dönüş Değeri

Bir sınıf örneğine tutturulmuş grafiksel nesnenin seçili olup olmama durumu. Tutturulmuş herhangi bir nesne yoksa 'false' dönüşü yapar.

Selected (Set Method)

Grafik nesnesinin seçili durumda olup olmadığını gösteren bayrağı ayarlar.

```
bool  Selected(
   bool  selected      // yeni bayrak değeri
   )
```

Parametreler

selected

[in] Bayrağın yeni değeri.

Dönüş Değeri

Başarılı ise 'true', bayrak değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::Selected için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- grafik nesnesinin "seçili" bayrağını al
   bool selected_flag=object.Selected();
   if(selected_flag)
     {
     //--- grafik nesnesinin "seçili" bayrağını ayarla
     object.Selected(false);
     }
  }
```

Selectable (Get Method)

Grafik nesnesinin seçilebilir olup olmadığını gösteren bayrağı alır Diğer bir deyişle nesne ya seçilebilir durumdadır ya da değildir.

```
bool  Selectable() const
```

Dönüş Değeri

Bir sınıf örneğine tutturulmuş grafiksel nesnenin seçilebilirliğini gösteren bayrak. Tutturulmuş herhangi bir nesne yoksa 'false' dönüşü yapar.

Selectable (Set Method)

Grafik nesnesinin seçilebilir olup olmadığını gösteren bayrağı ayarlar.

```
bool  Selectable(
   bool  selectable      // bayrak değeri
   )
```

Parametreler

selectable

[in] Bayrağın yeni değeri.

Dönüş Değeri

Başarılı ise 'true', bayrak değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::Selectable için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- grafik nesnesinin "seçilebilir" bayrağını al
   bool selectable_flag=object.Selectable();
   if(selectable_flag)
     {
     //--- grafik nesnesinin "seçilebilir" bayrağını ayarla
     object.Selectable(false);
     }
  }
```

Description (Get Method)

Belirtilen grafik nesnesinin açıklama metnini alır.

```
string  Description() const
```

Dönüş Değeri

Bir sınıf örneğine tutturulmuş grafiksel nesnenin açıklama metni. Tutturulmuş bir nesne yoksa NULL dönüşü yapar.

Description (Set Method)

Belirtilen grafik nesnesinin açıklama metnini ayarlar.

```
bool  Description(
   string  text      // metin
   )
```

Parametreler

text

[in] Yeni açıklama metni.

Dönüş Değeri

Başarılı ise 'true', açıklama değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::Description için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- grafiksel nesnenin açıklamasını al
   string description=object.Description();
   if(description=="")
     {
      //--- grafiksel nesnenin açıklamasını ayarla
      object.Description("MyObject");
     }
  }
```

Tooltip (Get Method)

Grafik nesnesinin araç-ipucu metnini alır.

```
string  Tooltip() const
```

Dönüş değeri

Bir sınıf örneğine tutturulmuş grafik nesnesinin araç-ipucu metni. Tutturulmuş bir nesne yoksa NULL dönüşü yapar.

Tooltip (Set Method)

Grafik nesnesinin araç-ipucu metnini ayarlar.

```
bool  Tooltip(
   string  new_tooltip      // yeni araç ipucu metni
   )
```

Parametreler

new_tooltip

[in] Araç-ipucu için yeni metin.

Dönüş değeri

Başarılı ise 'true', araç ipucu metni değiştirilemezse 'false'.

Not:

Araç ipucu ayarlanmamışsa, nesne aktive edildiğinde terminal tarafından otomatik olarak oluşturulacaktır. Araç ipucu, "\n" (satır sonu) değerinin ayarlanmasıyla devre-dışı bırakılabilir.

Timeframes (Get Method)

Grafik nesnesinin görünürlük bayraklarını alır.

```
int  Timeframes() const
```

Dönüş Değeri

Bir sınıf örneğine tutturulmuş grafiksel nesnenin görünürlük bayrakları. Tutturulmuş bir nesne yoksa 0 dönüşü yapar.

Timeframes (Set Method)

Grafik nesnesinin görünürlük bayraklarını ayarlar.

```
bool  Timeframes(
   int  new_timeframes      // görünürlük bayrakları
   )
```

Parametreler

new_timeframes

[in] Grafik nesnesinin yeni görünürlük bayrakları.

Dönüş Değeri

Başarılı ise 'true', görünürlük bayrağı değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::Timeframes için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- nesnenin görüntüleneceği periyotları al
   int timeframes=object.Timeframes();
   if(!(timeframes&OBJ_PERIOD_H1))
     {
      //--- nesnenin görüntüleneceği periyotları ayarla
      object.Timeframes(timeframes|OBJ_PERIOD_H1);
     }
  }
```

Z_Order (Get Method)

Grafiksel nesnenin çizelge üzerindeki tıklama önceliği değerini alır (CHARTEVENT_CLICK).

```
long  Z_Order() const
```

Dönüş Değeri

Sınıf örneğine atanmış grafik nesnesinin tıklama önceliği. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

Z_Order (Set Method)

Grafiksel nesnenin çizelge üzerindeki tıklama önceliği değerini ayarlar (CHARTEVENT_CLICK).

```
bool  Z_Order(
   long  value      // yeni özellik değeri
   )
```

Parametreler

value

[in] Grafiksel nesnenin çizelge üzerindeki tıklama önceliği için yeni değer (CHARTEVENT_CLICK).

Dönüş Değeri

Başarılı ise 'true', öncelik değeri değiştirilemezse 'false'.

Not

Z_Order, çizelge üzerindeki tıklama olayları (CHARTEVENT_CLICK) için grafik nesnesinin önceliğini temsil eder. Bu değeri sıfırdan (ön-tanımlı değerden) daha büyük bir sayıya ayarlayarak nesne önceliğini artırabilirsiniz.

CreateTime

Grafik nesnesinin oluşturulma zamanını alır.

```
datetime  CreateTime() const
```

Dönüş Değeri

Bir sınıf örneğine tutturulmuş grafiksel nesnenin oluşturulma zamanı. Tutturulmuş bir nesne yoksa 0 dönüşü yapar.

Örnek:

```
//--- CChartObject::CreateTime için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- grafik nesnesinin oluşturulma zamanını al
   datetime create_time=object.CreateTime();
  }
```

LevelsCount (Get Method)

Grafik nesnesindeki seviyelerin sayısını alır.

```
int  LevelsCount() const
```

Dönüş Değeri

Bir sınıf örneğine tutturulmuş grafiksel nesnedeki seviyelerin sayısı. Tutturulmuş bir nesne yoksa 0 dönüşü yapar.

LevelsCount (Set Method)

Grafik nesnesindeki seviyelerin sayısını ayarlar.

```
bool  LevelsCount(
   int  levels      // Seviyelerin sayısı
   )
```

Parametreler

levels

[in] Grafik nesnesi seviyelerinin yeni sayısı.

Dönüş Değeri

Başarılı ise 'true', seviye sayısı değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::LevelsCount için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- grafik nesnesinin seviyelerinin sayısını al
   int levels_count=object.LevelsCount();
   //--- grafik nesnesinin seviyelerinin sayısını ayarla
   object.LevelsCount(levels_count+1);
  }
```

LevelColor (Get Method)

Grafik nesnesindeki belirtilen seviyenin çizgi rengini alır.

```
color  LevelColor(
   int  level      // seviye numarası
   ) const
```

Parametreler

level

[in] Seviyenin numarası.

Dönüş Değeri

Bir sınıf örneğine tutturulmuş grafiksel nesnenin belirtilen seviyesinin çizgi rengi. Tutturulmuş bir nesne yoksa veya belirtilen seviye yoksa CLR_NONE değerine dönüş yapar.

LevelColor (Set Method)

Nesnenin belirtilen seviyesinin çizgi rengini ayarlar.

```
bool  LevelColor(
   int    level,          // seviye numarası
   color  new_color       // yeni renk
   )
```

Parametreler

level

[in] Seviyenin numarası.

new_color

[in] Nesnenin belirtilen seviyesi için yeni çizgi rengi.

Dönüş Değeri

Başarılı ise 'true', renk değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::LevelColor için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- grafik nesnesinin seviye rengini al
      color level_color=object.LevelColor(i);
      if(level_color!=clrRed)
```

```
         {
          //--- grafik nesnesinin seviye rengini ayarla
          object.LevelColor(i,clrRed);
         }
      }
   }
```

LevelStyle (Get Method)

Grafik nesnesindeki belirtilen seviyenin çizgi stilini alır.

```
ENUM_LINE_STYLE  LevelStyle(
   int  level      // seviye numarası
   ) const
```

Parametreler

level

[in] Seviyenin numarası.

Dönüş Değeri

Bir sınıf örneğine tutturulmuş grafiksel nesnenin belirtilen seviyesinin çizgisi stili. Tutturulmuş bir nesne yoksa veya belirtilen seviye yoksa WRONG_VALUE değerine dönüş yapar.

LevelStyle (Set Method)

Grafik nesnesindeki belirtilen seviyenin çizgi stilini ayarlar.

```
int  LevelStyle(
   int               level,      // seviye numarası
   ENUM_LINE_STYLE   style       // çizgi stili
   )
```

Parametreler

level

[in] Seviyenin numarası.

style

[in] Belirtilen seviyenin yeni çizgi stili.

Dönüş Değeri

Başarılı ise 'true', stil değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::LevelStyle için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- grafik nesnesindeki seviyenin çizgi stilini al
      ENUM_LINE_STYLE level_style=object.LevelStyle(i);
      if(level_style!=STYLE_SOLID)
```

```
         {
          //--- grafik nesnesindeki seviyenin çizgi stilini ayarla
          object.LevelStyle(i,STYLE_SOLID);
         }
      }
   }
```

LevelWidth (Get Method)

Grafik nesnesindeki belirtilen seviyenin çizgi kalınlığını alır.

```
int  LevelWidth(
   int  level      // seviye numarası
   ) const
```

Parametreler

level

[in] Seviyenin numarası.

Dönüş Değeri

Bir sınıf örneğine tutturulmuş grafiksel nesnenin belirtilen seviyesinin çizgisi kalınlığı. Tutturulmuş bir nesne yoksa veya belirtilen seviye yoksa -1 değerine dönüş yapar.

LevelWidth (Set Method)

Nesnenin belirtilen seviyesinin çizgi kalınlığını ayarlar.

```
bool  LevelWidth(
   int  level,          // seviye numarası
   int  new_width       // yeni kalınlık değeri
   )
```

Parametreler

level

[in] Seviyenin numarası.

new_width

[in] Nesnenin belirtilen seviyesi için yeni çizgi kalınlığı.

Dönüş Değeri

Başarılı ise 'true', kalınlık değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::LevelWidth için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- grafik nesnesinin seviyesinin çizgi kalınlığını al
      int level_width=object.LevelWidth(i);
      if(level_width!=1)
```

```
         {
          //--- grafik nesnesinin seviyesinin çizgi kalınlığını ayarla
          object.LevelWidth(i,1);
         }
      }
   }
```

LevelValue (Get Method)

Grafik nesnesindeki belirtilen seviyenin değerini alır.

```
double  LevelValue(
   int  level      // seviye numarası
   ) const
```

Parametreler

level

[in] Seviyenin numarası.

Dönüş Değeri

Bir sınıf örneğine tutturulmuş grafik nesnesindeki belirtilen seviyenin değeri. Tutturulmuş bir nesne yoksa veya belirtilen seviye yoksa EMPTY_VALUE değerine dönüş yapar.

LevelValue (Set Method)

Grafik nesnesindeki belirtilen seviyenin değerini ayarlar.

```
bool  LevelValue(
   int     level,          // seviye numarası
   double  new_value       // yeni değer
   )
```

Parametreler

level

[in] Seviyenin numarası.

new_value

[in] Belirtilen seviyenin yeni değeri.

Dönüş Değeri

Başarılı ise 'true', değer değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::LevelValue için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- grafik nesnesinin seviye değerini al
      double level_value=object.LevelValue(i);
      if(level_value!=0.1*i)
```

```
         {
          //--- grafik nesnesinin seviye değerini ayarla
          object.LevelValue(i,0.1*i);
         }
      }
   }
```

LevelDescription (Get Yöntemi)

Belirtilen grafik nesnesi seviyesinin açıklama metnini alır.

```
string  LevelDescription(
   int  level       // seviye numarası
   ) const
```

Parametreler

level

[in] Grafik nesnesi seviyesinin numarası.

Dönüş Değeri

Bir sınıf örneğine tutturulmuş grafiksel nesnenin seviyesinin açıklama metni. Tutturulmuş bir nesne yoksa veya belirtilen seviye yoksa NULL değerine dönüş yapar.

LevelDescription (Set Yöntemi)

Belirtilen grafik nesnesi seviyesinin açıklama metnini ayarlar.

```
bool  LevelDescription(
   int     level ,     // seviye numarası
   string  text        // metin
   )
```

Parametreler

level

[in] Grafik nesnesi seviyesinin numarası.

text

[in] Belirtilen grafik nesnesi seviyesinin açıklama metni için yeni değer.

Dönüş Değeri

Başarılı ise 'true', açıklama değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::LevelDescription için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //---
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- grafik nesnesi seviyesinin açıklamasını al
      string level_description=object.LevelDescription(i);
      if(level_description=="")
```

```
         {
          //--- grafik nesnesi seviyesinin açıklamasını ayarla
          object.LevelDescription(i,"Level_"+IntegerToString(i));
         }
      }
   }
```

GetInteger

Sınıf örneğine atanmış tamsayı (bool, char, uchar, short, ushort, int, uint, long, ulong, datetime veya color tipli) grafik özelliği değerlerinin alınabilmesi için, MQL5 API ObjectGetInteger() fonksiyonlarına kolay erişim sağlar. Fonksiyonun iki çağrı çeşidi vardır:

Doğruluk kontrolü yapılmadan özellik değerinin alınması

```
long  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,          // tamsayı özelliğin tanımlayıcısı
   int                           modifier=-1       // değiştirici
   ) const
```

Parametreler

prop_id

[in] Tamsayı grafik özelliğinin tanımlayıcısı.

modifier=-1

[in] Tamsayı özelliğin değiştiricisi (indisi).

Dönüş Değeri

Başarı durumunda tamsayı tipli özelliğin değerine, aksi durumda 0 değerine dönüş yapar.

Sonuç doğrulama yöntemini kullanarak özellik değerinin alınması

```
bool  GetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER  prop_id,       // tamsayı özelliğin tanımlayıcısı
   int                           modifier,      // değiştirici
   long&                         value          // çıkış değeri referansı
   ) const
```

Parametreler

prop_id

[in] Tamsayı değerli grafik nesnesi özelliğinin tanımlayıcısı.

modifier

[in] Tamsayı özelliğin değiştiricisi (indisi).

value

[out] Tamsayı tipli özellik değerini alacak olan referans değişkeni.

Dönüş Değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Örnek:

```
//--- CChartObject::GetInteger için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
```

```
{
 CChartObject object;
 //--- grafik nesnesinin rengini kolay yolla al
 printf("Nesnenin rengi: %s",ColorToString(object.GetInteger(OBJPROP_COLOR),true));
 //--- grafik nesnesinin rengini klasik yolla al
 long color_value;
 if(!object.GetInteger(OBJPROP_COLOR,0,color_value))
   {
    printf("Özellik değeri alınamadı, hata: %d",GetLastError());
    return;
   }
 else
    printf("Nesnenin rengi: %s",color_value);
 for(int i=0;i<object.LevelsCount();i++)
   {
    //--- seviye çizgisi kalınlığını kolay yöntemle al
    printf("Seviye %d kalınlığı: %d",i,object.GetInteger(OBJPROP_LEVELWIDTH,i));
    //--- seviye çizgisi kalınlığını klasik yöntemle al
    long width_value;
    if(!object.GetInteger(OBJPROP_LEVELWIDTH,i,width_value))
      {
       printf("Özellik alınamadı, hata: %d",GetLastError());
       return;
      }
    else
       printf("Seviye %d kalınlığı: %d",i,width_value);
   }
}
```

SetInteger

Sınıf örneğine atanmış tamsayı (bool, char, uchar, short, ushort, int, uint, long, ulong, datetime veya color tipli) grafik özelliği değerlerinin ayarlanabilmesi için, MQL5 API ObjectSetInteger() fonksiyonlarına kolay erişim sağlar. Fonksiyonun iki çağrı çeşidi vardır:

Değiştirici gerektirmeyen bir özellik değerinin ayarlanması

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,      // özellik tanımlayıcısı
   long                           value         // yeni değer
   )
```

Parametreler

prop_id

[in] Tamsayı değerli grafik nesnesi özelliğinin tanımlayıcısı.

value

[in] Özellik için yeni tamsayı değeri.

Değiştirici gerektiren bir özellik değerinin ayarlanması

```
bool  SetInteger(
   ENUM_OBJECT_PROPERTY_INTEGER   prop_id,      // özellik tanımlayıcısı
   int                            modifier,     // değiştirici
   long                           value         // yeni değer
   )
```

Parametreler

prop_id

[in] Tamsayı değerli grafik nesnesi özelliğinin tanımlayıcısı.

modifier

[in] Tamsayı özelliğin değiştiricisi (indisi).

value

[in] Özellik için yeni tamsayı değeri.

Dönüş Değeri

Başarılı ise 'true', özellik değeri değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::SetInteger için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- grafik nesnesi için yeni renk ayarla
```

```
   if(!object.SetInteger(OBJPROP_COLOR,clrRed))
     {
      printf("Özellik değeri ayarlama hatası %d",GetLastError());
      return;
     }
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- seviye genişliklerini ayarla
      if(!object.SetInteger(OBJPROP_LEVELWIDTH,i,i))
        {
         printf("Özellik değeri ayarlama hatası %d",GetLastError());
         return;
        }
     }
  }
```

GetDouble

Sınıf örneğine atanmış çift-duyarlı (float veya double tipli) grafik özelliği değerlerinin alınabilmesi için, MQL5 API ObjectGetDouble() fonksiyonlarına kolay erişim sağlar. Fonksiyonun iki çağrı çeşidi vardır:

Doğruluk kontrolü yapılmadan özellik değerinin alınması

```
double  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,          // çift-duyarlı özelliğin tanımlayıcı
   int                          modifier=-1      // değiştirici
   ) const
```

Parametreler

prop_id

[in] Çift-duyarlı grafik özelliğinin tanımlayıcısı.

modifier=-1

[in] Çift-duyarlı özelliğin değiştiricisi (indisi).

Dönüş Değeri

Başarı durumunda istenilen özelliğin değerine, aksi durumda EMPTY_VALUE değerine dönüş yapar.

Sonuç doğrulama yöntemini kullanarak özellik değerinin alınması

```
bool  GetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE  prop_id,       // özellik tanımlayıcısı
   int                          modifier,      // değiştirici
   double&                      value          // çıkış değeri referansı
   ) const
```

Parametreler

prop_id

[in] Çift-duyarlı grafik özelliğinin tanımlayıcısı.

modifier

[in] Çift-duyarlı özelliğin değiştiricisi (indisi).

value

[out] Çift-duyarlı özellik değerini alacak olan referans değişkeni.

Dönüş Değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Örnek:

```
//--- CChartObject::GetDouble için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
```

```
  CChartObject object;
  //---
  for(int i=0;i<object.LevelsCount();i++)
    {
     //--- seviye değerlerini basit yöntem ile al
     printf("Seviye %d değeri=%f",i,object.GetDouble(OBJPROP_LEVELVALUE,i));
     //--- seviye değerlerini klasik yöntemle al
     double value;
     if(!object.SetDouble(OBJPROP_LEVELVALUE,i,value))
       {
        printf("çift-duyarlı özellik değeri alınamadı, hata %d",GetLastError());
        return;
       }
     else
        printf("Seviye %d değeri=%f",i,value);
    }
}
```

SetDouble

Sınıf örneğine atanmış çift-duyarlı (double veya float) grafik özelliği değerlerinin değiştirilebilmesi için, MQL5 API ObjectSetDouble() fonksiyonlarına kolay erişim sağlar. Fonksiyonun iki çağrı çeşidi vardır:

Değiştirici gerektirmeyen bir özellik değerinin ayarlanması

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,      // Özellik tanımlayıcısı
   double                        value         // Yeni değer
   )
```

Parametreler

prop_id

[in] Çift-duyarlı grafik özelliğinin tanımlayıcısı.

value

[in] Çift-duyarlı grafik özelliği için yeni değer.

Değiştirici gerektiren bir özellik değerinin ayarlanması

```
bool  SetDouble(
   ENUM_OBJECT_PROPERTY_DOUBLE   prop_id,       // özellik tanımlayıcısı
   int                           modifier,      // değiştirici
   double                        value          // yeni değer
   )
```

Parametreler

prop_id

[in] Çift-duyarlı grafik özelliğinin tanımlayıcısı.

modifier

[in] Çift-duyarlı özelliğin değiştiricisi (indisi).

value

[in] Çift-duyarlı grafik özelliği için yeni değer.

Dönüş Değeri

Başarılı ise 'true', özellik değeri değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::SetDouble için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
//---
```

```
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- grafik nesnesinin seviye değerini ayarla
      if(!object.SetDouble(OBJPROP_LEVELVALUE,i,0.1*i))
        {
         printf("Özellik değeri ayarlanamadı, hata: %d",GetLastError());
         return;
        }
     }
  }
```

GetString

Sınıf örneğine atanmış dizgi biçimindeki grafik özelliği değerlerinin alınabilmesi için, MQL5 API ObjectGetString() fonksiyonlarına kolay erişim sağlar. Fonksiyonun iki çağrı çeşidi vardır:

Doğruluk kontrolü yapılmadan özellik değerinin alınması

```
string  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,          // dizgi özelliğin tanımlayıcısı
   int                          modifier=-1       // değiştirici
   ) const
```

Parametreler

prop_id

[in] Dizgi değerli grafik nesnesi özelliğinin tanımlayıcısı.

modifier=-1

[in] Dizgi özelliğin değiştiricisi (indisi).

Dönüş Değeri

Dizgi özelliğin değeri.

Sonuç doğrulama yöntemini kullanarak özellik değerinin alınması

```
bool  GetString(
   ENUM_OBJECT_PROPERTY_STRING  prop_id,       // dizgi özelliğin tanımlayıcısı
   int                          modifier,      // değiştirici
   string&                      value          // çıkış değeri referansı
   ) const
```

Parametreler

prop_id

[in] Dizgi değerli grafik nesnesi özelliğinin tanımlayıcısı.

modifier

[in] Dizgi özelliğin değiştiricisi (indisi).

value

[out] Dizgi özelliğin değerini alacak olan referans değişkeni.

Dönüş Değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Örnek:

```
//--- CChartObject::GetString için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
```

```
   CChartObject object;
   string       value;
   //--- grafik nesnesinin ismini kolay yolla al
   printf("Nesnenin ismi: '%s'",object.GetString(OBJPROP_NAME));
   //--- grafik nesnesinin ismini klasik yolla al
   if(!object.GetString(OBJPROP_NAME,0,value))
     {
      printf("Özellik değeri alınamadı, hata %d",GetLastError());
      return;
     }
   else
      printf("Nesnenin ismi: '%s'",value);
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- seviye açıklamasını kolay yöntemle al
      printf("Seviye %d açıklaması: '%s'",i,object.GetString(OBJPROP_LEVELTEXT,i));
      //--- seviye açıklamasını klasik yöntemle al
      if(!object.GetString(OBJPROP_LEVELTEXT,i,value))
        {
         printf("Özellik değeri alınamadı, hata %d",GetLastError());
         return;
        }
      else
         printf("Seviye %d açıklaması: '%s'",i,value);
     }
  }
```

SetString

Sınıf örneğine atanmış dizgi biçimindeki grafik özelliği değerlerinin alınabilmesi için, MQL5 API ObjectSetString() fonksiyonlarına kolay erişim sağlar. Fonksiyonun iki çağrı çeşidi vardır:

Değiştirici gerektirmeyen bir özellik değerinin ayarlanması

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,      // özelliğin tanımlayıcısı
   string                        value         // yeni değer
   )
```

Parametreler

prop_id

[in] Dizgi değerli grafik nesnesi özelliğinin tanımlayıcısı.

value

[in] Dizgi biçimli özellik için yeni değer.

Değiştirici gerektiren bir özellik değerinin ayarlanması

```
bool  SetString(
   ENUM_OBJECT_PROPERTY_STRING   prop_id,       // özelliğin tanımlayıcısı
   int                           modifier,      // değiştirici
   string                        value          // yeni değer
   )
```

Parametreler

prop_id

[in] Dizgi değerli grafik nesnesi özelliğinin tanımlayıcısı.

modifier

[in] Dizgi özelliğin değiştiricisi (indisi).

value

[in] Dizgi biçimli özellik için yeni değer.

Dönüş Değeri

Başarılı ise 'true', özellik değeri değiştirilemezse 'false'.

Örnek:

```
//--- CChartObject::SetString için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- grafik nesnesinin ismini değiştir
   if(!object.SetString(OBJPROP_NAME,"MyObject"))
```

```
     {
      printf("Özellik ayarlanamadı, hata %d",GetLastError());
      return;
     }
   for(int i=0;i<object.LevelsCount();i++)
     {
      //--- seviye açıklamalarını değiştir
      if(!object.SetString(OBJPROP_LEVELTEXT,i,"Level_"+IntegerToString(i)))
        {
         printf("Özellik ayarlanamadı, hata %d",GetLastError());
         return;
        }
     }
  }
```

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CChartObject::Save için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CChartObject object=new CChartObject;
   //--- nesne parametrelerini ayarla
   //--- . . .
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Save(file_handle))
        {
         //--- dosya kaydetme hatası
         printf("Dosya kaydedilemedi. Hata %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
  }
```

Load

Grafik nesnesinin parametrelerini dosyadan yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan dosyanın tanıtıcı değeri.

Dönüş Değeri

Başarıyla tamamlanmışsa 'true', aksi durumda 'false'.

Örnek:

```
//--- CChartObject::Load için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   int           file_handle;
   CChartObject object;
   //--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!object.Load(file_handle))
        {
         //--- dosya yükleme hatası
         printf("Dosya yükleme başarısız. Hata %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
   //--- nesneyi kullan
   //--- . . .
  }
```

Type

Grafik nesnesinin tip tanımlayıcısını alır.

```
virtual int  Type() const
```

Dönüş Değeri

Nesne tipi tanımlayıcısı (CChartObject için 0x8888).

Örnek:

```
//--- CChartObject::Type için bir örnek
#include <ChartObjects\ChartObject.mqh>
//---
void OnStart()
  {
   CChartObject object;
   //--- nesne tipini al
   int type=object.Type();
  }
```

Çizgi Nesnenelri

"Çizgiler" grubundan grafik nesneleri.

Bu bölüm, "Çizgiler" grubundan grafiksel nesnelerin sınıflarıyla çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için açıklamalar içermektedir.

Sınıf ismi	Nesne
CChartObjectVLine	"Dikey çizgi" grafik nesnesi
CChartObjectHLine	"Yatay çizgi" grafik nesnesi
CChartObjectTrend	"Trend çizgisi" grafik nesnesi
CChartObjectTrendByAngle	"Açılı trend çizgisi" grafik nesnesi
CChartObjectCycles	"Döngüsel çizgiler" grafik nesnesi

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

CChartObjectVLine

CChartObjectVLine sınıfı, "Dikey çizgi" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectVLine sınıfı, "Dikey çizgi" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectVLine : public CChartObject
```

Başlık

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectVLine

Sınıf Yöntemleri

Oluştur	
Create	"Dikey çizgi" grafik nesnesini oluşturur.
Girdi/çıktı	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Dikey çizgi" grafik nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge alt-penceresi
   datetime  time           // zaman koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time

[in] Tutturma noktasının zaman koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesine ait nesne tipi tanımlayıcısına dönüş yapar.

```
int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectVLine için OBJ_VLINE).

CChartObjectHLine

CChartObjectHLine sınıfı, "Yatay çizgi" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectHLine sınıfı, "Yatay çizgi" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectHLine : public CChartObject
```

Başlık

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectHLine

Sınıf Yöntemleri

Oluştur	
Create	"Yatay çizgi" grafik nesnesini oluşturur.
Girdi/çıktı	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Yatay çizgi" grafik nesnesini oluşturur.

```
bool  Create(
   long    chart_id,      // Çizelge tanımlayıcısı
   string  name,          // Nesne ismi
   long    window,        // Çizelge penceresi
   double  price          // Fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

price

[in] Tutturma noktasının fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesine ait nesne tipi tanımlayıcısına dönüş yapar.

```
int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectHLine için OBJ_HLINE).

CChartObjectTrend

CChartObjectTrend sınıfı, "Trend çizgisi" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectTrend sınıfı, "Trend çizgisi" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectTrend : public CChartObject
```

Başlık

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Kalıtım hiyerarşisi

CObject

CChartObject

CChartObjectTrend

İlk nesil

CChartObjectChannel, CChartObjectFibo, CChartObjectFiboChannel, CChartObjectFiboExpansion, CChartObjectGannFan, CChartObjectGannGrid, CChartObjectPitchfork, CChartObjectRegression, CChartObjectStdDevChannel, CChartObjectTrendByAngle

Sınıf Yöntemleri

Oluştur	
Create	"Trend çizgisi" grafik nesnesini oluşturur.
Özellikler	
RayLeft	"Sol ışın" özelliğinin değerinin alır/ayarlar
RayRight	"Sağ ışın" özelliğinin değerinin alır/ayarlar
Girdi/çıktı	
virtual Save	Sanal dosya yazım yöntemi
virtual Load	Sanal dosya okuma yöntemi
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CChartObject

Sınıftan türetilen yöntemler 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

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Trend çizgisi" grafik nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge alt-penceresi
   datetime  time1,         // 1-inci zaman koordinatı coordinate
   double    price1,        // 1-inci fiyat koordinatı
   datetime  time2,         // 2-inci zaman koordinatı
   double    price2         // 1-inci fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

price2

[in] İkinci tutturma noktasının fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

RayLeft (Get Yöntemi)

"Sol ışın" özelliğinin değerini alır.

```
bool  RayLeft() const
```

Dönüş değeri

Sınıf örneğine atanan "Sol ışın" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

RayLeft (Set Yöntemi)

"Sol ışın" özelliği için yeni değer ayarlar.

```
bool  RayLeft(
   bool  ray      // yeni bayrak değeri
   )
```

Parametreler

ray

[in] "Sol ışın" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', bayrak değişmediyse 'false'.

RayRight (Get Yöntemi)

"Sağ ışın" özelliğinin değerini alır.

```
bool  RayRight() const
```

Dönüş değeri

Sınıf örneğine atanan "Sağ ışın" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

RayRight (Set Yöntemi)

"Sağ ışın" özelliği için yeni değer ayarlar.

```
bool  RayRight(
   bool  ray      // bayrak
   )
```

Parametreler

ray

[in] "Sağ ışın" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', bayrak değişmediyse 'false'.

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Dosyadan nesne parametreleri yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesine ait nesne tipi tanımlayıcısına dönüş yapar.

```
int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectTrend için OBJ_TREND).

CChartObjectTrendByAngle

CChartObjectTrendByAngle sınıfı, "Açılı trend çizgisi" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectTrendByAngle sınıfı, "Açılı trend çizgisi" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectTrendByAngle : public CChartObjectTrend
```

Başlık

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Kalıtım hiyerarşisi

CObject
CChartObject
CChartObjectTrend
CChartObjectTrendByAngle

İlk nesil

CChartObjectGannLine

Sınıf Yöntemleri

Oluştur	
Create	"Açılı trend çizgisi" grafik nesnesini oluşturur
Özellikler	
Angle	"Açı" özelliğinin değerini alır/ayarlar
Girdi/çıktı	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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,

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Sınıftan türetilen yöntemler CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Açılı trend çizgisi" grafik nesnesini oluşturur

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   long      window,        // çizelge penceresi
   datetime  time1,         // 1-inci zaman koordinatı coordinate
   double    price1,        // 1-inci fiyat koordinatı
   datetime  time2,         // 2-inci zaman koordinatı
   double    price2         // 1-inci fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

price2

[in] İkinci tutturma noktasının fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Angle (Get Yöntemi)

"Açı" özelliğinin değerini alır.

```
double  Angle() const
```

Dönüş değeri

Sınıf örneğine atanan "Açı" özelliğinin değeri. Nesne atanmamışsa EMPTY_VALUE değerine dönüş yapar.

Angle (Set Yöntemi)

"Açı" özelliği için yeni değer ayarlar.

```
bool  Angle(
   double  angle      // Açı
   )
```

Parametreler

angle

[in] "Açı" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Type

Grafik nesnesine ait nesne tipi tanımlayıcısına dönüş yapar.

```
int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectTrendByAngle için OBJ_TRENDBYANGLE).

CChartObjectCycles

CChartObjectCycles sınıfı, "Döngüsel çizgiler" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectCycles sınıfı, "Döngüsel çizgiler" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectCycles : public CChartObject
```

Başlık

```
#include <ChartObjects\ChartObjectsLines.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectCycles

Sınıf Yöntemleri

Oluştur	
Create	"Döngüsel çizgiler" grafik nesnesini oluşturur
Girdi/çıktı	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Döngüsel çizgiler" grafik nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   long      window,        // çizelge penceresi
   datetime  time1,         // 1-inci zaman koordinatı coordinate
   double    price1,        // 1-inci fiyat koordinatı
   datetime  time2,         // 2-inci zaman koordinatı
   double    price2         // 1-inci fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

price2

[in] İkinci tutturma noktasının fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesine ait nesne tipi tanımlayıcısına dönüş yapar.

```
int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectCycles için OBJ_CYCLES).

Kanal Nesneleri

"Kanallar" grubundan grafik nesneleri.

Bu bölüm, "Kanallar" grubundan grafiksel nesnelerin sınıflarıyla çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için açıklamalar içermektedir.

Sınıf ismi	Nesne
CChartObjectChannel	"Eşit-uzaklık Kanalı" grafik nesnesi
CChartObjectRegression	"Lineer Regresyon Kanalı" grafik nesnesi
CChartObjectStdDevChannel	"Standart Sapma Kanalı" grafik nesnesi
CChartObjectPitchfork	"Andrew Çatalı" grafik nesnesi

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

CChartObjectChannel

CChartObjectChannel sınıfı, "Eşit-uzaklık Kanalı" grafik nesnesinin özelliklerine kolay erişim sağlar.

Açıklama

CChartObjectChannel, "Eşit-uzaklık Kanalı" nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectChannel : public CChartObjectTrend
```

Başlık

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectChannel

Sınıf Yöntemleri

Oluştur	
Create	"Eşit-uzaklık Kanalı" nesnesini oluşturur
Girdi/çıktı	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Eşit-uzaklık Kanalı" nesnesini oluşturur

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge alt-penceresi
   datetime  time1,         // ilk tutturma noktasının zaman koordinatı
   double    price1,        // ilk tutturma noktasının fiyat koordinatı
   datetime  time2,         // ikinci tutturma noktasının zaman koordinatı
   double    price2,        // ikinci tutturma noktasının fiyat koordinatı
   datetime  time3,         // üçüncü tutturma noktasının zaman koordinatı
   double    price3         // üçüncü tutturma noktasının zaman koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

price2

[in] İkinci tutturma noktasının fiyat koordinatı.

time3

[in] Üçüncü tutturma noktasının zaman koordinatı.

price3

[in] Üçüncü tutturma noktasının fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesine ait nesne tipi tanımlayıcısına dönüş yapar.

```
int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectChannel için OBJ_CHANNEL).

CChartObjectRegression

CChartObjectRegression sınıfı, "Lineer Regresyon Kanalı" grafik nesnesinin özelliklerine kolay erişim sağlamak amacıyla geliştirilmiştir.

Açıklama

CChartObjectRegression, "Lineer Regresyon Kanalı" nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectRegression : public CChartObjectTrend
```

Başlık

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectRegression

Sınıf Yöntemleri

Oluştur	
Create	"Lineer Regresyon Kanalı" nesnesini oluşturur
Girdi/çıktı	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Lineer Regresyon Kanalı" nesnesini oluşturur

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   long      window,        // çizelge penceresi
   datetime  time1,         // ilk zaman koordinatı
   datetime  time2          // ikinci zaman koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesine ait nesne tipi tanımlayıcısına dönüş yapar.

```
int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectRegression için OBJ_REGRESSION).

CChartObjectStdDevChannel

CChartObjectStdDevChannel sınıfı, "Standart Sapma Kanalı" grafik nesnesinin özelliklerine kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CChartObjectStdDevChannel sınıfı, "Standart Sapma Kanalı" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectStdDevChannel : public CChartObjectTrend
```

Başlık

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectStdDevChannel

Sınıf Yöntemleri

Oluştur	
Create	"Standart Sapma Kanalı" grafik nesnesini oluşturur
Özellikler	
Deviations	"Sapma" özelliğinin değerini alır/ayarlar
Girdi/çıktı	
virtual Save	Sanal dosya yazım yöntemi
virtual Load	Sanal dosya okuma yöntemi
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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,

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Sınıftan türetilen yöntemler CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Standart Sapma Kanalı" grafik nesnesini oluşturur

```
bool  Create(
   long      chart_id,      // Çizelge tanımlayıcısı
   string    name,          // Nesne ismi
   int       window,        // Çizelge alt-penceresi
   datetime  time1,         // Birinci zaman koordinatı
   datetime  time2,         // İkinci zaman koordinatı
   double    deviation      // Sapma
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

deviation

[in] "Sapma" özelliği için sayısal değer.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Deviation (Get Yöntemi)

"Sapma" özelliğinin sayısal değerini alır.

```
double  Deviation() const
```

Dönüş değeri

Sınıf örneğine atanan "Sapma" özelliğinin sayısal değeri. Nesne atanmamışsa EMPTY_VALUE değerine dönüş yapar.

Deviation (Set Yöntemi)

"Sapma" özelliğinin sayısal değerini ayarlar.

```
bool  Deviation(
   double  deviation      // yeni sapma değeri
   )
```

Parametreler

deviation

[in] "Sapma" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Dosyadan nesne parametreleri yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesine ait nesne tipi tanımlayıcısına dönüş yapar.

```
int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectStdDevChannel için OBJ_STDDEVCHANNEL).

CChartObjectPitchfork

CChartObjectPitchfork sınıfı, "Andrew Çatalı" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectPitchfork, "Andrew Çatalı" grafik nesnesinin özelliklerine kolay erişim sağlar.

Bildirim

```
class CChartObjectPitchfork : public CChartObjectTrend
```

Başlık

```
#include <ChartObjects\ChartObjectsChannels.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectPitchfork

Sınıf Yöntemleri

Oluştur	
Create	"Andrew Çatalı"" nesnesini oluşturur
Girdi/çıktı	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Andrew Çatalı"" nesnesini oluşturur

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   long      window,        // çizelge penceresi
   datetime  time1,         // ilk tutturma noktasının zaman koordinatı
   double    price1,        // ilk tutturma noktasının fiyat koordinatı
   datetime  time2,         // ikinci tutturma noktasının zaman koordinatı
   double    price2,        // ikinci tutturma noktasının fiyat koordinatı
   datetime  time3,         // üçüncü tutturma noktasının zaman koordinatı
   double    price3         // üçüncü tutturma noktasının fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci fiyat noktasının zaman koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

price2

[in] İkinci tutturma noktasının fiyat koordinatı.

time3

[in] Üçüncü tutturma noktasının zaman koordinatı.

price3

[in] Üçüncü tutturma noktasının fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesine ait nesne tipi tanımlayıcısına dönüş yapar.

```
int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectPitchfork için OBJ_PITCHFORK).

Gann Araçları

"Gann Araçları" grubundan grafik nesneleri.

Bu bölüm, "Gann Araçları" grubundan grafiksel nesnelerin sınıflarıyla çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için açıklamalar içermektedir.

Sınıf ismi	Nesne
CChartObjectGannLine	"Gann Çizgisi" grafik nesnesi
CChartObjectGannFan	"Gann Yelpazesi" grafik nesnesi
CChartObjectGannGrid	"Gann Izgarası" grafik nesnesi

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

CChartObjectGannLine

CChartObjectGannLine sınıfı, "Gann Çizgisi" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectGannLine sınıfı, "Gann Çizgisi" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectGannLine : public CChartObjectTrendByAngle
```

Başlık

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Kalıtım hiyerarşisi

CObject
CChartObject
CChartObjectTrend
CChartObjectTrendByAngle
CChartObjectGannLine

Sınıf Yöntemleri

Create	
Create	"Gann Çizgisi" grafik nesnesini oluşturur
Özellikler	
PipsPerBar	"Ölçek" özelliğinin değerini alır/atarlar
Girdi/çıktı	
virtual Save	Sanal dosya yazım yöntemi
virtual Load	Sanal dosya okuma yöntemi
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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,

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Sınıftan türetilen yöntemler CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Sınıftan türetilen yöntemler CChartObjectTrendByAngle

Angle, Angle, Create

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Gann Çizgisi" grafik nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge alt-penceresi
   datetime  time1,         // ilk zaman koordinatı
   double    price1,        // ilk fiyat koordinatı
   datetime  time2,         // ikinci zaman koordinatı
   double    ppb            // çubuk başı pip değeri
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

ppb

[in] Çubuk başına düşen pip değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

PipsPerBar (Get Yöntemi)

"Çubuk başına pip" özelliğinin değerini alır.

```
double  PipsPerBar() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Çubuk başına pip" özelliğinin değeri. Nesne atanmamışsa EMPTY_VALUE değerine dönüş yapar.

PipsPerBar (Set Yöntemi)

"Çubuk başına pip" özelliğinin değerini ayarlar.

```
bool  PipsPerBar(
   double  ppb      // Çubuk başına düşen pip
   )
```

Parametreler

ppb

[in] "Çubuk başına pip" özelliğinin yeni değeri

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Dosyadan nesne parametreleri yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısını alır.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectGannLine için OBJ_GANNLINE).

CChartObjectGannFan

CChartObjectGannFan sınıfı, "Gann Yelpazesi" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectGannFan sınıfı, "Gann Yelpazesi" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectGannFan : public CChartObjectTrend
```

Başlık

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectGannFan

Sınıf Yöntemleri

Oluştur	
Create	"Gann Yelpazesi" grafik nesnesini oluşturur
Özellikler	
PipsPerBar	"Çubuk başına pip" özelliğinin değerini alır/ayarlar
Downtrend	"Aşağı-trend" özelliğinin değerini alır/ayarlar
Girdi/çıktı	
virtual Save	Sanal dosya yazım yöntemi
virtual Load	Sanal dosya okuma yöntemi
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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,

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Sınıftan türetilen yöntemler CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Gann Yelpazesi" grafik nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge alt-penceresi
   datetime  time1,         // ilk zaman koordinatı
   double    price1,        // ilk fiyat koordinatı
   datetime  time2,         // ikinci zaman koordinatı
   double    ppb            // çubuk başı pip değeri
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

ppb

[in] Çubuk başına düşen pip değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

PipsPerBar (Get Yöntemi)

"Çubuk başına pip" özelliğinin değerini alır.

```
double  PipsPerBar() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Çubuk başına pip" özelliğinin değeri. Nesne atanmamışsa EMPTY_VALUE değerine dönüş yapar.

PipsPerBar (Set Yöntemi)

"Çubuk başına pip" özelliğinin değerini ayarlar.

```
bool  PipsPerBar(
   double  ppb      // Çubuk başına düşen pip
   )
```

Parametreler

ppb

[in] "Çubuk başına pip" özelliğinin yeni değeri

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Downtrend (Get Yöntemi)

"Aşağı-trend" özelliğinin değerini alır.

```
bool  Downtrend() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Aşağı-trend" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

Downtrend (Set Yöntemi)

"Aşağı-trend" özelliği için yeni değer ayarlar.

```
bool  Downtrend(
   bool  downtrend      // bayrak değeri
   )
```

Parametreler

downtrend

[in] "Aşağı-trend" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Dosyadan nesne parametreleri yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısını alır.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectGannFan için OBJ_GANNFAN).

CChartObjectGannGrid

CChartObjectGannGrid sınıfı, "Gann Izgarası" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectGannGrid sınıfı, "Gann Izgarası" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectGannGrid : public CChartObjectTrend
```

Başlık

```
#include <ChartObjects\ChartObjectsGann.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectGannGrid

Sınıf Yöntemleri

Oluştur	
Create	Creates graphic object "Gann Grid"
Özellikler	
PipsPerBar	"Çubuk başına pip" özelliğinin değerini alır/ayarlar
Downtrend	"Aşağı-trend" özelliğinin değerini alır/ayarlar
Girdi/çıktı	
virtual Save	Sanal dosya yazım yöntemi
virtual Load	Sanal dosya okuma yöntemi
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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,

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Sınıftan türetilen yöntemler CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Gann Izgarası" grafik nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge alt-penceresi
   datetime  time1,         // ilk zaman koordinatı
   double    price1,        // ilk fiyat koordinatı
   datetime  time2,         // ikinci zaman koordinatı
   double    ppb            // çubuk başı pip değeri
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

ppb

[in] Çubuk başına düşen pip değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

PipsPerBar (Get Yöntemi)

"Çubuk başına pip" özelliğinin değerini alır.

```
double  PipsPerBar() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Çubuk başına pip" özelliğinin değeri. Nesne atanmamışsa EMPTY_VALUE değerine dönüş yapar.

PipsPerBar (Set Yöntemi)

"Çubuk başına pip" özelliğinin değerini ayarlar.

```
bool  PipsPerBar(
   double  ppb      // Çubuk başına düşen pip
   )
```

Parametreler

ppb

[in] "Çubuk başına pip" özelliğinin yeni değeri

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Downtrend (Get Yöntemi)

"Aşağı-trend" özelliğinin değerini alır.

```
bool  Downtrend() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Aşağı-trend" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

Downtrend (Set Yöntemi)

"Aşağı-trend" özelliği için yeni değer ayarlar.

```
bool  Downtrend(
   bool  downtrend      // bayrak değeri
   )
```

Parametreler

downtrend

[in] "Aşağı-trend" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Dosyadan nesne parametreleri yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısını alır.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectGannGrid için OBJ_GANNGRID).

Fibonacci Araçları

"Fibonacci Araçları" grubundan grafik nesneleri.

Bu bölüm, "Fibonacci araçları" grubundan grafiksel nesnelerin sınıflarıyla çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için açıklamalar içermektedir.

Sınıf ismi	Nesne
CChartObjectFibo	"Fibonacci Trend-dışı Hareketi" grafik nesnesi
CChartObjectFiboTimes	"Fibonacci Zaman Dilimleri" grafik nesnesi
CChartObjectFiboFan	"Fibonacci Yelpazesi" Grafik nesnesi
CChartObjectFiboArc	"Fibonacci Yayı" grafik nesnesi
CChartObjectFiboChannel	"Fibonacci Kanalı" grafik nesnesi
CChartObjectFiboExpansion	"Fibonacci Açılımı" grafik nesnesi

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

CChartObjectFibo

CChartObjectFibo sınıfı, "Fibonacci Trend-dışı Hareketi" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectFibo sınıfı, "Fibonacci Trend-dışı Hareketi" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectFibo : public CChartObjectTrend
```

Başlık

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFibo

Sınıf Yöntemleri

Oluştur	
Create	"Fibonacci Trend-dışı Hareketi" grafik nesnesini oluşturur
Girdi/çıktı	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Fibbonacci Trend-dışı Hareketi" nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge alt-penceresi
   datetime  time1,         // ilk zaman koordinatı
   double    price1,        // ilk fiyat koordinatı
   datetime  time2,         // ikinci zaman koordinatı
   double    price2         // ikinci fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

price2

[in] İkinci tutturma noktasının fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectFibo için OBJ_FIBO).

CChartObjectFiboTimes

CChartObjectFiboTimes sınıfı, "Fibonacci Zaman-Dilimleri" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectFiboTimes sınıfı, "Fibonacci Zaman-Dilimleri" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectFiboTimes : public CChartObject
```

Başlık

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Kalıtım hiyerarşisi

CObject

CChartObject

CChartObjectFiboTimes

Sınıf Yöntemleri

Oluştur	
Create	Creates graphic object "Fibonacci Time Zones"
Girdi/çıktı	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Fibonacci Zaman Dilimleri" grafik nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge alt-penceresi
   datetime  time1,         // ilk zaman koordinatı
   double    price1,        // ilk fiyat koordinatı
   datetime  time2,         // ikinci zaman koordinatı
   double    price2         // ikinci fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

price2

[in] İkinci tutturma noktasının fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectFiboTimes için OBJ_FIBOTIMES).

CChartObjectFiboFan

CChartObjectFiboFan sınıfı, "Fibonacci Yelpazesi" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectFiboFan sınıfı, "Fibonacci Yelpazesi" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectFiboFan : public CChartObject
```

Başlık

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectFiboFan

Sınıf Yöntemleri

Oluştur	
Create	"Fibonacci Yelpazesi" grafik nesnesini oluşturur
Girdi/çıktı	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Fibonacci Yelpazesi" grafik nesnesini oluşturur .

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge alt-penceresi
   datetime  time1,         // ilk zaman koordinatı
   double    price1,        // ilk fiyat koordinatı
   datetime  time2,         // ikinci zaman koordinatı
   double    price2         // ikinci fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

price2

[in] İkinci tutturma noktasının fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectFiboFan için OBJ_FIBOFAN).

CChartObjectFiboArc

CChartObjectFiboArc, "Fibonacci Yayı" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiş bir sınıftır.

Açıklama

CChartObjectFiboArc sınıfı "Fibonacci Yayı" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectFiboArc : public CChartObject
```

Başlık

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectFiboArc

Sınıf Yöntemleri

Oluştur	
Create	"Fibonacci Yayı" nesnesini oluşturur
Özellikler	
Scale	"Ölçek" özelliğinin değerini alır/atarlar
Ellipse	"Elips" özelliğinin değerini alır/ayarlar
Girdi/çıktı	
virtual Save	Sanal dosya yazım yöntemi
virtual Load	Sanal dosya okuma yöntemi
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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,

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Fibonacci Yayı" nesnesini oluşturur

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge alt-penceresi
   datetime  time1,         // ilk zaman koordinatı
   double    price1,        // ilk fiyat koordinatı
   datetime  time2,         // ikinci zaman koordinatı
   double    price2,        // ikinci fiyat koordinatı
   double    scale          // ölçek
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

price2

[in] İkinci tutturma noktasının fiyat koordinatı.

scale

[in] Ölçek.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Scale (Get Yöntemi)

"Ölçek" özelliğinin değerini alır.

```
double  Scale() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Ölçek" özelliğinin değeri. Nesne atanmamışsa EMPTY_VALUE değerine dönüş yapar.

Scale (Set Yöntemi)

"Ölçek" özelliği için yeni değer ayarlar.

```
bool  Scale(
   double  scale      // ölçek
   )
```

Parametreler

scale

[in] "Ölçek" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Ellipse (Get Yöntemi)

"Elips" özelliğinin değerini alır.

```
bool  Ellipse() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Elips" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

Ellipse (Set Yöntemi)

"Elips" özelliğinin değeri için yeni bayrak ayarlar.

```
bool  Ellipse(
   bool  ellipse      // bayrak değeri
   )
```

Parametreler

ellipse

[in] "Elips" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Dosyadan nesne parametreleri yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tiipi tanımlayıcı (CChartObjectFiboArc için OBJ_FIBOARC).

CChartObjectFiboChannel

CChartObjectFiboChannel sınıfı, "Fibonacci Kanalı" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectFiboChannel sınıfı, "Fibonacci Kanalı" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectFiboChannel : public CChartObjectTrend
```

Başlık

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFiboChannel

Sınıf Yöntemleri

Oluştur	
Create	"Fibonacci Kanalı" grafik nesnesini oluşturur
Girdi/çıktı	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Fibonacci Kanalı" nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge alt-penceresi
   datetime  time1,         // ilk zaman koordinatı
   double    price1,        // ilk fiyat koordinatı
   datetime  time2,         // ikinci zaman koordinatı
   double    price2,        // ikinci fiyat koordinatı
   datetime  time3,         // üçüncü zaman koordinatı
   double    price3         // üçüncü fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

price2

[in] İkinci tutturma noktasının fiyat koordinatı.

time3

[in] Üçüncü tutturma noktasının zaman koordinatı.

price3

[in] Üçüncü tutturma noktasının fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectFiboChannel için OBJ_FIBOCHANNEL).

CChartObjectFiboExpansion

CChartObjectFiboExpansion sınıfı, "Fibonacci Açılımı" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectFiboExpansion sınıfı, "Fibonacci Açılımı" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectFiboExpansion : public CChartObjectTrend
```

Başlık

```
#include <ChartObjects\ChartObjectsFibo.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectTrend
      CChartObjectFiboExpansion

Sınıf Yöntemleri

Oluştur	
Create	"Fibonacci Açılımı" grafik nesnesini oluşturur
Girdi/çıktı	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CChartObjectTrend

RayLeft, RayLeft, RayRight, RayRight, Create, Save, Load

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Fibonacci Açılımı" grafik nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge alt-penceresi
   datetime  time1,         // ilk zaman koordinatı
   double    price1,        // ilk fiyat koordinatı
   datetime  time2,         // ikinci zaman koordinatı
   double    price2,        // ikinci fiyat koordinatı
   datetime  time3,         // üçüncü zaman koordinatı
   double    price3         // üçüncü fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

price2

[in] İkinci tutturma noktasının fiyat koordinatı.

time3

[in] Üçüncü tutturma noktasının zaman koordinatı.

price3

[in] Üçüncü tutturma noktasının fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

(CChartObjectFiboExpansion için OBJ_EXPANSION).

Elliott Araçları

"Elliot Araçları" grubundan grafik nesneleri.

Bu bölüm, "Elliot Araçları" grubundan grafiksel nesnelerin sınıflarıyla çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için açıklamalar içermektedir.

Sınıf ismi	Nesne
CChartObjectElliottWave3	"Düzeltici dalga" grafik nesnesi
CChartObjectElliottWave5	Dürtü dalgası grafik nesnesi

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

CChartObjectElliottWave3

CChartObjectElliottWave3 sınıfı, "Düzeltici Dalga" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectElliottWae3 sınıfı, "Düzeltici Dalga" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectElliottWave3 : public CChartObject
```

Başlık

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

Kalıtım hiyerarşisi

CObject

CChartObject

CChartObjectElliottWave3

İlk nesil

CChartObjectElliottWave5

Sınıf Yöntemleri

Oluştur	
Create	"Düzeltici Dalga" grafik nesnesini oluşturur
Özellikler	
Degree	"Derece" özelliğinin değerini alır/ayarlar
Lines	"Çizgiler" özelliğinin değerini alır/ayarlar
Girdi/çıktı	
virtual Save	Sanal dosya yazım yöntemi
virtual Load	Sanal dosya okuma yöntemi
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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,

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

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

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Düzeltici Dalga" isimli grafik nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge alt-penceresi
   datetime  time1,         // ilk zaman koordinatı
   double    price1,        // ilk fiyat koordinatı
   datetime  time2,         // ikinci zaman koordinatı
   double    price2,        // ikinci fiyat koordinatı
   datetime  time3,         // üçüncü zaman koordinatı
   double    price3         // üçüncü fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

price2

[in] İkinci tutturma noktasının fiyat koordinatı.

time3

[in] Üçüncü tutturma noktasının zaman koordinatı.

price3

[in] Üçüncü tutturma noktasının zaman koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Degree (Get Yöntemi)

"Derece" özelliğinin değerini alır.

```
ENUM_ELLIOT_WAVE_DEGREE  Degree() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Derece" özelliğinin değeri. Atanmış hiçbir nesne yoksa WRONG_VALUE dönüşü yapar.

Degree (Set Yöntemi)

"Derece" özelliğinin değerini ayarlar.

```
bool  Degree(
   ENUM_ELLIOT_WAVE_DEGREE  degree      // özellik değeri
   )
```

Parametreler

degree

[in] "Derece" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Lines (Get Yöntemi)

"Çizgiler" özelliğinin değerini alır.

```
bool  Lines() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Çizgiler" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

Lines (Set Yöntemi)

"Çizgiler" özelliği için yeni değer ayarlar.

```
bool  Lines(
   bool  lines      // yeni değer
   )
```

Parametreler

lines

[in] "Çizgiler" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', bayrak değişmediyse 'false'.

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Dosyadan nesne parametreleri yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectElliottWave3 için OBJ_ELLIOTWAVE3).

CChartObjectElliottWave5

CChartObjectElliottWave5 sınıfı, "Dürtü Dalgası" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectElliottWave5 sınıfı, "Dürtü Dalgası" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectElliottWave5 : public CChartObjectElliottWave3
```

Başlık

```
#include <ChartObjects\ChartObjectsElliott.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectElliottWave3
      CChartObjectElliottWave5

Sınıf Yöntemleri

Oluştur	
Create	"Dürtü Dalgası" nesnesi oluşturur
Girdi/çıktı	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CChartObjectElliottWave3

Degree, Degree, Lines, Lines, Create, Save, Load

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Dürtü Dalgası" grafik nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge alt-penceresi
   datetime  time1,         // ilk zaman koordinatı
   double    price1,        // ilk fiyat koordinatı
   datetime  time2,         // ikinci zaman koordinatı
   double    price2,        // ikinci fiyat koordinatı
   datetime  time3,         // üçüncü zaman koordinatı
   double    price3,        // Üçüncü fiyat koordinatı
   datetime  time4,         // dördüncü zaman koordinatı
   double    price4,        // dördüncü fiyat koordinatı
   datetime  time5          // beşinci zaman koordinatı
   double    price5,        // beşinci fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

price2

[in] İkinci tutturma noktasının fiyat koordinatı.

time3

[in] Üçüncü tutturma noktasının zaman koordinatı.

price3

[in] Üçüncü tutturma noktasının fiyat koordinatı.

time4

[in] Dördüncü tutturma noktasının zaman koordinatı.

price4

[in] Dördüncü tutturma noktasının fiyat koordinatı.

time5

[in] Beşinci tutturma noktasının zaman koordinatı.

price5

[in] Beşinci tutturma noktasının fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (OBJ_ELLIOTWAVE5CChartObjectElliottWave5 için OBJ_ELLIOTWAVE5).

Objects Shapes

"Şekiller" grubundan grafik nesneleri.

Bu bölüm, "Şekiller" grubundan grafiksel nesnelerin sınıflarıyla çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için açıklamalar içermektedir.

Sınıf ismi	Nesne
CChartObjectRectangle	"Dikdörtgen" grafik nesnesi
CChartObjectTriangle	"Üçgen" grafik nesnesi
CChartObjectEllipse	"Elips" grafik nesnesi

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

CChartObjectRectangle

CChartObjectRectangle sınıfı, "Dikdörtgen" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectRectangle sınıfı, "Dikdörtgen" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectRectangle : public CChartObject
```

Başlık

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectRectangle

Sınıf Yöntemleri

Oluştur	
Create	"Dikdörtgen" grafik nesnesini oluşturur
Girdi/çıktı	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Dikdörtgen" grafik nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   long      window,        // çizelge penceresi
   datetime  time1,         // ilk zaman koordinatı
   double    price1,        // ilk fiyat koordinatı
   datetime  time2,         // ikinci zaman koordinatı
   double    price2         // ikinci fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

price2

[in] İkinci tutturma noktasının fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesine ait nesne tipi tanımlayıcısına dönüş yapar.

```
int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectRectangle için OBJ_RECTANGLE).

CChartObjectTriangle

CChartObjectTriangle sınıfı, "Üçgen" grafik nesnesinin özelliklerine kolay erişim için düzenlenmiş bir sınıftır.

Açıklama

CChartObjectTriangle sınıfı, "Üçgen" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectTriangle : public CChartObject
```

Başlık

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectTriangle

Sınıf Yöntemleri

Oluştur	
Create	"Üçgen" grafik nesnesini oluşturur
Girdi/çıktı	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Üçgen" grafik nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   long      window,        // çizelge penceresi
   datetime  time1,         // ilk zaman koordinatı
   double    price1,        // ilk fiyat koordinatı
   datetime  time2,         // ikinci zaman koordinatı
   double    price2,        // ikinci fiyat koordinatı
   datetime  time3,         // üçüncü zaman koordinatı
   double    price3         // üçüncü fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

price2

[in] İkinci tutturma noktasının fiyat koordinatı.

time3

[in] Üçüncü tutturma noktasının zaman koordinatı.

price3

[in] Üçüncü tutturma noktasının fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesine ait nesne tipi tanımlayıcısına dönüş yapar.

```
int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectTriangle için OBJ_TRIANGLE).

CChartObjectEllipse

CChartObjectEllipse sınıfı, "Elips" grafik nesnesinin özelliklerine kolay erişim için düzenlenmiş bir sınıftır.

Açıklama

CChartObjectEllipse sınıfı, "Elips" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectEllipse : public CChartObject
```

Başlık

```
#include <ChartObjects\ChartObjectsShapes.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectEllipse

Sınıf Yöntemleri

Oluştur	
Create	"Elips" grafik nesnesini oluşturur
Girdi/çıktı	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

Create

"Elips" grafik nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge alt-penceresi
   datetime  time1,         // ilk zaman koordinatı
   double    price1,        // ilk fiyat koordinatı
   datetime  time2,         // ikinci zaman koordinatı
   double    price2,        // ikinci fiyat koordinatı
   datetime  time3,         // üçüncü zaman koordinatı
   double    price3         // üçüncü fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time1

[in] Birinci tutturma noktasının zaman koordinatı.

price1

[in] Birinci tutturma noktasının fiyat koordinatı.

time2

[in] İkinci tutturma noktasının zaman koordinatı.

price2

[in] İkinci tutturma noktasının fiyat koordinatı.

time3

[in] Üçüncü tutturma noktasının zaman koordinatı.

price3

[in] Üçüncü tutturma noktasının fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesine ait nesne tipi tanımlayıcısına dönüş yapar.

```
int  Type() const
```

Dönüş değeri

ONesne tipi tanımlayıcısı (CChartObjectEllipse için OBJ_ELLIPSE).

Objects Arrows

Ok türündeki grafik nesneleri için bir grup.

Bu bölüm, "Ok" grubundan grafiksel nesnelerin sınıflarıyla çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için açıklamalar içermektedir. Tanımlamak gerekirse, ok, kullanıcıya gösterilen ve belli bir kod ile eşleşen bir simgedir. "Ok" grafik nesnesinin çizelgede simge görüntülemek için kullanılabilecek iki türü bulunmaktadır:

- "Ok" nesnesi, görüntülenen simgenin kodunu belirlemenizi sağlar.
- Belli simge tiplerini görüntülemeye yarayan (ve belli sabit kodlara gelen) grup nesneleri.

Rassal kodlu simgeler gösteren ok için bir sınıf

Sınıf ismi	Ok nesnesinin ismi
CChartObjectArrow	Ok

Sabit kodlu ok simgeleri için sınıflar

Sınıf ismi	Ok nesnesinin ismi
CChartObjectArrowCheck	Onay
CChartObjectArrowDown	Yukarı ok
CChartObjectArrowUp	Aşağı ok
CChartObjectArrowStop	Dur işareti
CChartObjectArrowThumbDown	Başparmak yukarı
CChartObjectArrowThumbUp	Başparmak aşağı
CChartObjectArrowLeftPrice	Sol fiyat etiketi
CChartObjectArrowRightPrice	Sağ fiyat etiketi

Ayrıca bakınız

Nesne tipleri, Nesne bağlama yöntemleri, Grafik nesneleri

CChartObjectArrow

CChartObjectArrow sınıfı, "Ok" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectArrow, soyundan gelen tüm sınıflar için "Ok" grafik nesnesinin genel özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectArrow : public CChartObject
```

Başlık

```
#include <ChartObjects\ChartObjectsArrows.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectArrow

İlk nesil

CChartObjectArrowCheck, CChartObjectArrowDown, CChartObjectArrowLeftPrice, CChartObjectArrowRightPrice, CChartObjectArrowStop, CChartObjectArrowThumbDown, CChartObjectArrowThumbUp, CChartObjectArrowUp

Sınıf Yöntemleri

Oluştur	
Create	"Ok" grafik nesnesini oluşturur
Özellikler	
ArrowCode	Ok nesnesinin karakter kodunu alır/ayarlar
Anchor	"Çapa" (tutturucu) özelliğinin türünü alır/ayarlar
Girdi/çıktı	
virtual Save	Sanal dosya yazım yöntemi
virtual Load	Sanal dosya okuma yöntemi
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CChartObject

Sınıftan türetilen yöntemler 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

Ayrıca bakınız

Nesne tipleri, Nesne bağlama yöntemleri, Grafik nesneleri

Create

Bir "ok" nesnesi oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge penceresi
   datetime  time,          // zaman koordinatı
   double    price,         // fiyat koordinatı
   char      code           // ok kodu
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcı (mevcut çizelge için 0).

name

[in] Nesne ismi (benzersiz olmalı).

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time

[in] Zaman koordinatı.

price

[in] Fiyat koordinatı.

code

[in] "Ok" nesnesinin kodu.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'

Örnek:

```
//--- CChartObjectArrow::Create için bir örnek
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
//--- nesne parametrelerini ayarla
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- ok oluşturma hatası
      printf("Ok oluşturulamadı: Hata %d!",GetLastError());
      //---
```

```
      return;
     }
//--- oku kullan
//--- . . .
  }
```

ArrowCode (Get Yöntemi)

"Ok" nesnesinin karakter kodunu alır.

```
char  ArrowCode() const
```

Dönüş değeri

Sınıf örneğine bağlanmış "Ok" nesnesinin karakter kodu. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

ArrowCode (Set Yöntemi)

"Ok" nesnesinin karakter kodunu ayarlar

```
bool  ArrowCode(
   char  code      // kod değeri
   )
```

Parametreler

code

[in] Nesnenin karakter kodu için yeni değer.

Dönüş değeri

Başarılı ise 'true', kod değeri değişmediyse 'false'.

Örnek:

```
//--- CChartObjectArrow::ArrowCode için bir örnek
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   char              code=181;
//--- nesne parametrelerini ayarla
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,code))
     {
      //--- ok oluşturma hatası
      printf("Ok oluşturulamadı: Hata %d!",GetLastError());
      //---
      return;
     }
//--- oku muhtemel kod değişiklikleri için tara
//--- . . .
//--- ok kodunu al
   if(arrow.ArrowCode()!=code)
     {
     //--- ok kodunu ayarla
```

```
      arrow.ArrowCode(code);
      }
//--- oku kullan
//--- . . .
   }
```

Anchor (Get Method)

"Ok" nesnesinin çapa (tutturucu) türünü alır

```
ENUM_ARROW_ANCHOR  Anchor() const
```

Dönüş değeri

Sınıf örneğine atanmış "Ok" nesnesinin çapa türü. Atanmış hiçbir nesne yoksa WRONG_VALUE dönüşü yapar.

Anchor (Set Method)

"Ok" nesnesinin çapa türünü ayarlar

```
bool  Anchor(
   ENUM_ARROW_ANCHOR  anchor      // yeni çapa türü
   )
```

Parametreler

anchor

[in] Yeni çapa türü

Dönüş değeri

Başarılı ise 'true', çapa türü değişmediyse 'false'.

Örnek:

```
//--- CChartObject::Anchor için bir örnek
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   ENUM_ARROW_ANCHOR anchor=ANCHOR_BOTTOM;
//--- nesne parametrelerini ayarla
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- ok oluşturma hatası
      printf("Ok oluşturulamadı: Hata %d!",GetLastError());
      //---
      return;
     }
//--- okun çapa türünü al
   if(arrow.Anchor()!=anchor)
     {
     //--- ok çapasını ayarla
     arrow.Anchor(anchor);
     }
```

```
//--- oku kullan
//--- . . .
  }
```

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Örnek:

```
//--- CChartObjectArrow::Save için bir örnek
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   int               file_handle;
   CChartObjectArrow arrow;
//--- nesne parametrelerini ayarla
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"Arrow",0,TimeCurrent(),price,181))
     {
      //--- ok oluşturma hatası
      printf("Ok oluşturulamadı: Hata %d!",GetLastError());
      //---
      return;
     }
//--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_WRITE|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!arrow.Save(file_handle))
        {
         //--- dosya kaydetme hatası
         printf("Dosya kaydedilemedi: Hata %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
  }
```

Load

Dosyadan nesne parametreleri yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Örnek:

```
//--- CChartObjectArrow::Load için bir örnek
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   int               file_handle;
   CChartObjectArrow arrow;
//--- dosyayı aç
   file_handle=FileOpen("MyFile.bin",FILE_READ|FILE_BIN|FILE_ANSI);
   if(file_handle>=0)
     {
      if(!arrow.Load(file_handle))
        {
         //--- dosya yükleme hatası
         printf("Dosya yüklenemedi: Hata %d!",GetLastError());
         FileClose(file_handle);
         //---
         return;
        }
      FileClose(file_handle);
     }
//--- oku kullan
//--- . . .
  }
```

Type

Grafik nesnesinin tipini alır.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (örneğin CChartObjectArrow için OBJ_ARROW)

Örnek:

```
//--- CChartObjectArrow::Type için bir örnek
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
   CChartObjectArrow arrow;
   //--- ok tipini al
   int type=arrow.Type();
  }
```

Sabit kodlu oklar

"Sabit kodlu ok" sınıfları aşağıda verilen grafik nesnesi özelliklerine kolay erişim sağlamak için tasarlanmıştır:

Sınıf ismi	Ok nesnesi ismi
CChartObjectArrowCheck	"Onay Oku"
CChartObjectArrowDown	"Aşağı Ok"
CChartObjectArrowUp	"Yukarı Ok"
CChartObjectArrowStop	"Dur Oku"
CChartObjectArrowThumbDown	"Olumlu" ("Başparmak yukarı")
CChartObjectArrowThumbUp	"Olumsuz" ("Başparmak aşağı")
CChartObjectArrowLeftPrice	"Sol fiyat etiketi" oku
CChartObjectArrowRightPrice	"Sağ fiyat etiketi" oku

Açıklama

"Sabit kodlu ok" sınıfları ilgili nesne özelliklerine erişim sağlar.

Bildirimler

```
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;
```

Başlık

```
<ChartObjects\ChartObjectsArrows.mqh>
```

Sınıf Yöntemleri

Oluşturma	
Create	Belirtilen grafik nesnesini oluşturur
Özellikler	
ArrowCode	Kod değişimleri için bir "Saplama" (stub) yöntemi
Girdi/çıktı	
virtual Type	Sanal tanımlama yöntemi

Ayrıca bakınız

Nesne tipleri, Nesne bağlama yöntemleri, Grafik nesneleri

Create

"Sabit kodlu ok" grafik nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge penceresi
   datetime  time,          // zaman koordinatı
   double    price          // fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time

[in] Zaman koordinatı.

price

[in] Fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Örnek:

```
//--- CChartObjectArrowCheck::Create için bir örnek
//--- CChartObjectArrowDown::Create için bir örnek
//--- CChartObjectArrowUp::Create için bir örnek
//--- CChartObjectArrowStop::Create için bir örnek
//--- CChartObjectArrowThumbDown::Create için bir örnek
//--- CChartObjectArrowThumbUp::Create için bir örnek
//--- CChartObjectArrowLeftPrice::Create için bir örnek
//--- CChartObjectArrowRightPrice::Create için bir örnek
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- örnek olarak CChartObjectArrowCheck sınıfını alalım
   CChartObjectArrowCheck arrow;
//--- nesne parametrelerini ayarla
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
```

```
      {
       //--- ok oluşturma hatası
       printf("Ok oluşturma hatası, %d!",GetLastError());
       //---
       return;
      }
//--- oku kullan
//--- . . .
  }
```

ArrowCode

"Ok" nesnesi için kod değişimlerini yasaklar.

```
bool  ArrowCode(
   char  code      // kod değeri
   )
```

Parametreler

code

[in] Herhangi bir değer

Dönüş değeri

Daima 'false'.

Örnek:

```
//--- CChartObjectArrowCheck::ArrowCode
//--- CChartObjectArrowDown::ArrowCode için bir örnek
//--- CChartObjectArrowUp::ArrowCode için bir örnek
//--- CChartObjectArrowStop::ArrowCode için bir örnek
//--- CChartObjectArrowThumbDown::ArrowCode için bir örnek
//--- CChartObjectArrowThumbUp::ArrowCode için bir örnek
//--- CChartObjectArrowLeftPrice::ArrowCode için bir örnek
//--- CChartObjectArrowRightPrice::ArrowCode için bir örnek
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- örnek olarak CChartObjectArrowCheck sınıfını alalım
   CChartObjectArrowCheck arrow;
//--- nesne parametrelerini ayarla
   double price=SymbolInfoDouble(Symbol(),SYMBOL_BID);
   if(!arrow.Create(0,"ArrowCheck",0,TimeCurrent(),price))
     {
      //--- ok oluşturma hatası
      printf("Ok oluşturma hatası %d!",GetLastError());
      //---
      return;
     }
//--- ok kodunu ayarla
   if(!arrow.ArrowCode(181))
     {
      //--- bu bir hata değil
      printf("ok kodu değiştirilemiyor");
     }
//--- oku kullan
//--- . . .
  }
```

Type

Grafik nesnesinin tanımlayıcısına dönüş yapar

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectArrowCheck için OBJ_ARROW_CHECK, CChartObjectArrowDown için OBJ_ARROW_DOWN, CChartObjectArrowUp için OBJ_ARROW_UP, CChartObjectArrowStop için OBJ_ARROW_STOP, CChartObjectArrowThumbDown için OBJ_ARROW_THUMB_DOWN, CChartObjectArrowThumbUp için OBJ_ARROW_THUMB_UP, CChartObjectArrowLeftPrice için OBJ_ARROW_LEFT_PRICE, CChartObjectArrowRightPrice için OBJ_ARROW_RIGHT_PRICE).

Örnek:

```
//--- CChartObjectArrowCheck::Type için bir örnek
//--- CChartObjectArrowDown::Type için bir örnek
//--- CChartObjectArrowUp::Type için bir örnek
//--- CChartObjectArrowStop::Type için bir örnek
//--- CChartObjectArrowThumbDown::Type için bir örnek
//--- CChartObjectArrowThumbUp::Type için bir örnek
//--- CChartObjectArrowLeftPrice::Type için bir örnek
//--- CChartObjectArrowRightPrice::Type için bir örnek
#include <ChartObjects\ChartObjectsArrows.mqh>
//---
void OnStart()
  {
//--- örnek olarak CChartObjectArrowCheck sınıfını alalım
   CChartObjectArrowCheck arrow;
//--- ok tipini al
   int type=arrow.Type();
  }
```

Nesne Kontrolleri

"Nesne Kontrolleri" grubundan grafik nesneleri.

Bu bölüm, "Nesne Kontrolleri" grubundan grafiksel nesnelerin sınıflarıyla çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için açıklamalar içermektedir.

Sınıf ismi	Nesne
CChartObjectText	"Metin" grafik nesnesi
CChartObjectLabel	"Metin Etiketi" grafik nesnesi
CChartObjectEdit	"Düzen" grafik nesnesi
CChartObjectButton	"Düğme" grafik nesnesi
CChartObjectSubChart	"Çizelge" grafik nesnesi
CChartObjectBitmap	"Biteşlem" grafik nesnesi
CChartObjectBmpLabel	"Biteşlem Etiketi" grafik nesnesi
CChartObjectRectLabel	"Dikdörtgen Etiket" grafik nesnesi

Ayrıca bakınız

Nesne tipleri, Grafik nesneleri

CChartObjectText

CChartObjectText sınıfı, "Metin" grafik nesnesinin özelliklerine kolay erişim için düzenlenmiş bir sınıftır.

Açıklama

CChartObjectText sınıfı, "Metin" grafik nesnesinin özelliklerine kolay erişim sağlar.

Bildirim

```
class CChartObjectText : public CChartObject
```

Başlık

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Kalıtım hiyerarşisi

CObject

CChartObject

CChartObjectText

İlk nesil

CChartObjectLabel

Sınıf Yöntemleri

Oluştur	
Create	"Metin" grafik nesnesini oluşturur
Özellikler	
Angle	"Açı" özelliğinin değerini alır/ayarlar
Font	"Yazı-tipi" özelliğinin değerini alır/ayarlar
FontSize	"Yazı-tipi boyutu" özelliğinin değerini alır/ayarlar
Anchor	"Çapa" (tutturucu) özelliğinin türünü alır/ayarlar
Girdi/çıktı	
virtual Save	Sanal dosya yazım yöntemi
virtual Load	Sanal dosya okuma yöntemi
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CChartObject

Sınıftan türetilen yöntemler 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

Türetilmiş sınıflar:

- CChartObjectLabel

Ayrıca bakınız

Nesne tipleri, Nesne özellikleri, Nesne bağlama yöntemleri, Grafik nesneleri

Create

"Metin" grafik nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge alt-penceresi
   datetime  time,          // zaman koordinatı
   double    price          // fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time

[in] Tutturma noktasının zaman koordinatı.

price

[in] Tutturma noktasının fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Angle (Get Yöntemi)

"Açı" özelliğinin değerini alır.

```
double  Angle() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Açı" özelliğinin değeri. Nesne atanmamışsa EMPTY_VALUE değerine dönüş yapar.

Angle (Set Yöntemi)

"Açı" özelliği için yeni değer ayarlar.

```
bool  Angle(
   double  angle      // yeni açı
   )
```

Parametreler

angle

[in] "Açı" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Font (Get Yöntemi)

"Yazı-tipi" özelliğinin değerini alır.

```
string  Font() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Yazı-tipi" özelliğinin değeri. Atanmış bir nesne yoksa "" dönüşü yapar.

Font (Set Yöntemi)

"Yazı-tipi" özelliğinin değerini ayarlar.

```
bool  Font(
   string  font      // yeni yazı-tipi
   )
```

Parametreler

font

[in] "Yazı-tipi" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

FontSize (Get Yöntemi)

"Yazı-tipi boyutu" özelliğinin değerini alır.

```
int  FontSize() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Yazı-tipi boyutu" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

FontSize (Set Yöntemi)

"Yazı-tipi boyutu" özelliğinin değerini ayarlar.

```
bool  FontSize(
   int  size      // yeni yazı-tipi boyutu
   )
```

Parametreler

size

[in] "Yazı-tipi" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Anchor (Get Yöntemi)

"Çapa" (tutturucu) özelliğinin değerini alır.

```
ENUM_ANCHOR_POINT  Anchor() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Çapa" özelliğinin değeri. Atanmış hiçbir nesne yoksa WRONG_VALUE dönüşü yapar.

Anchor (Set Yöntemi)

"Çapa" (tutturucu) özelliğinin değerini ayarlar.

```
bool  Anchor(
   ENUM_ANCHOR_POINT  anchor      // yeni değer
   )
```

Parametreler

anchor

[in] "Çapa" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Dosyadan nesne parametreleri yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (OBJ_TEXTCChartObjectText için OBJ_TEXT).

CChartObjectLabel

CChartObjectLabel sınıfı, "Etiket" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectLabel sınıfı, "Etiket" grafik nesnesinin özelliklerine kolay erişim sağlar.

Bildirim

```
class CChartObjectLabel : public CChartObjectText
```

Başlık

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel

İlk nesil

CChartObjectEdit, CChartObjectRectLabel

Sınıf Yöntemleri

Oluştur	
Create	"Etiket" grafik nesnesini oluşturur
Özellikler	
X_Distance	"X-Uzaklığı" özelliğinin değerini alır/ayarlar
Y_Distance	"Y-Uzaklığı" özelliğinin değerini alır/ayarlar
X_Size	"X-Büyüklüğü" özelliğinin değerini alır/ayarlar
Y_Size	"Y-Büyüklüğü" özelliğinin değerini alır/ayarlar
Corner	"Köşe" özelliğinin değerini alır/ayarlar
Time	Fiyat koordinatındaki değişimler için bir "Saplama" (stub) yöntemi
Price	Zaman koordinatındaki değişimler için bir "Saplama" yöntemi
Girdi/çıktı	
virtual Save	Sanal dosya yazım yöntemi
virtual Load	Sanal dosya okuma yöntemi

Oluştur	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Ayrıca bakınız

Nesne tipleri, Nesne özellikleri, Çizelge köşesi, Nesne bağlama yöntemleri, Grafik nesneleri

Create

"Etiket" grafik nesnesini oluşturur.

```
bool  Create(
   long     chart_id,      // Çizelge tanımlayıcısı
   string   name,          // Nesne ismi
   int      window,        // Çizelge penceresi
   int      X,             // X koordinatı
   int      Y              // Y koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

X

[in] X koordinatı.

Y

[in] Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

X_Distance (Get Yöntemi)

"X Uzaklığı" özelliğinin değerini alır.

```
int  X_Distance() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "X Uzaklığı" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

X_Distance (Set Yöntemi)

"X Uzaklığı" özelliğinin değerini ayarlar.

```
bool  X_Distance(
   int  X      // yeni değer
   )
```

Parametreler

X

[in] "X Uzaklığı" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Y_Distance (Get Yöntemi)

"Y Uzaklığı" özelliğinin değerini alır.

```
int  Y_Distance() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Y Uzaklığı" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

Y_Distance (Set Yöntemi)

"Y Uzaklığı" özelliği için yeni değer ayarlar.

```
bool  Y_Distance(
   int  Y      // yeni değer
   )
```

Parametreler

Y

[in] "Y Uzaklığı" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

X_Size

"X Büyüklüğü" özelliğinin değerini alır.

```
int  X_Size() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "X Büyüklüğü" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

Y_Size

"Y Büyüklüğü" özelliğinin değerini alır.

```
int  Y_Size() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Y Büyüklüğü" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

Corner (Get Yöntemi)

"Köşe" özelliğinin değerini alır.

```
ENUM_BASE_CORNER  Corner() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Köşe" özelliğinin değeri. Atanmış hiçbir nesne yoksa WRONG_VALUE dönüşü yapar.

Corner (Set Yöntemi)

"Köşe" özelliği için yeni değer ayarlar.

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // yeni değer
   )
```

Parametreler

corner

[in] "Köşe" özelliği için yeni değer

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Time

Zaman koordinatının değiştirilmesini engeller.

```
bool  Time(
   datetime  time      // herhangi bir değer
   )
```

Parametreler

time

[in] datetime tipli herhangi bir değer.

Dönüş değeri

Daima 'false'.

Price

Fiyat koordinatının değiştirilmesini engeller.

```
bool  Price(
   double  price      // her hangi bir değer
   )
```

Parametreler

price

[in] double tipli herhangi bir değer.

Dönüş değeri

Daima 'false'.

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Dosyadan nesne parametreleri yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

(CChartObjectLabel için OBJ_LABEL).

CChartObjectEdit

CChartObjectEdit sınıfı, "Düzen" grafik nesnesinin özelliklerine kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CChartObjectEdit sınıfı, "Düzen" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectEdit : public CChartObjectLabel
```

Başlık

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Kalıtım hiyerarşisi

CObject
CChartObject
CChartObjectText
CChartObjectLabel
CChartObjectEdit

İlk nesil

CChartObjectButton

Sınıf Yöntemleri

Oluştur	
Create	"Düzen" grafik nesnesini oluşturur
Özellikler	
TextAlign	"Metin Hizalama" özelliğinin değerini alır/ayarlar
X_Size	"X Büyüklüğü" özelliğinin değerini alır
Y_Size	"Y Büyüklüğü" özelliğinin değerini alır
BackColor	"Arka-plan Rengi" özelliğinin değerini alır/ayarlar
BorderColor	"Kenarlık Rengi" özelliğinin değerini alır/ayarlar
ReadOnly	"Salt Okunur" özelliğinin değerini alır/ayarlar
Angle	"Açı" özelliğinin değerini alır/ayarlar
Girdi/çıktı	
virtual Save	Sanal dosya yazım yöntemi
virtual Load	Sanal dosya okuma yöntemi

Oluştur	
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Sınıftan türetilen yöntemler CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Ayrıca bakınız

Nesne tipleri, Nesne özellikleri, Çizelge köşesi, Nesne bağlama yöntemleri, Grafik nesneleri

Create

"Düzen" grafik nesnesini oluşturur.

```
bool  Create(
   long     chart_id,      // Çizelge tanımlayıcısı
   string   name,          // Nesne ismi
   int      window,        // Çizelge penceresi
   int      X,             // X koordinatı
   int      Y,             // Y koordinatı
   int      sizeX,         // X büyüklüğü
   int      sizeY          // Y büyüklüğü
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

X

[in] X koordinatı.

Y

[in] Y koordinatı.

sizeX

[in] X büyüklüğü.

sizeY

[in] Y büyüklüğü.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

TextAlign (Get Yöntemi)

"Metin Hizalama" özelliğinin değerini alır (metin hizalama modu).

```
ENUM_ALIGN_MODE  TextAlign() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Metin Hizalama" özelliğinin değeri.

TextAlign (Set Yöntemi)

"Metin Hizalama" özelliği için yeni değer ayarlar. (metin hizalama modu).

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // yeni değer
   )
```

Parametreler

align

[in] "Metin Hizalama" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

X_Size

"X Büyüklüğü" özelliğinin değerini ayarlar.

```
bool  X_Size(
   int  size      // yeni değer
   )
```

Parametreler

size

[in] "X Büyüklüğü" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Y_Size

"Y Büyüklüğü" özelliği için yeni değer ayarlar.

```
bool  Y_Size(
   int  size      // yeni değer
   )
```

Parametreler

size

[in] "Y Büyüklüğü" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

BackColor (Get Yöntemi)

"Araka-plan Rengi" özelliğinin değerini alır.

```
color  BackColor() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Araka-plan Rengi" özelliğinin değeri. Atanmış hiçbir nesne yoksa CLR_NONE değerine dönüş yapar.

BackColor (Set Yöntemi)

"Araka-plan Rengi" özelliğinin değerini ayarlar.

```
bool  BackColor(
   color  new_color      // yeni arka-plan rengi
   )
```

Parametreler

new_color

[in] "Araka-plan Rengi" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

BorderColor (Get Yöntemi)

"Kenarlık Rengi" özelliğinin değerini alır.

```
color  BorderColor() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Kenarlık Rengi" özelliğinin değeri. Atanmış hiçbir nesne yoksa CLR_NONE değerine dönüş yapar.

BorderColor (Set Yöntemi)

"Kenarlık Rengi" özelliğinin değerini ayarlar.

```
bool  BorderColor(
   color  new_color      // yeni kenarlık rengi
   )
```

Parametreler

new_color

[in] "Kenarlık Rengi" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

ReadOnly (Get Yöntemi)

"Salt Okunur" özelliğinin değerini alır.

```
bool  ReadOnly() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Salt Okunur" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

ReadOnly (Set Yöntemi)

"Salt Okunur" özelliğinin değerini ayarlar.

```
bool  ReadOnly(
   const  bool  flag      // yeni değer
   )
```

Parametreler

flag

[in] "Salt Okunur" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Angle

"Açı" özelliğinin değişmesini engeller.

```
bool  Angle(
   double  angle      // herhangi bir değer
   )
```

Parametreler

angle

[in] double tipli herhangi bir değer.

Dönüş değeri

Daima 'false'.

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Dosyadan nesne parametreleri yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectEdit için OBJ_EDIT).

CChartObjectButton

CChartObjectButton sınıfı, "Düğme" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectButton sınıfı, "Düğme" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectButton : public CChartObjectEdit
```

Başlık

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Kalıtım hiyerarşisi

CObject
CChartObject
CChartObjectText
CChartObjectLabel
CChartObjectEdit
CChartObjectButton

İlk nesil

CChartObjectPanel

Sınıf Yöntemleri

Oluştur	
Create	CChartObjectEdit sınıfından türetilmiştir
Özellikler	
State	Düğme durumunu (Basılı/Serbest) alır/ayarlar
Girdi/çıktı	
virtual Save	Sanal dosya yazım yöntemi
virtual Load	Sanal dosya okuma yöntemi
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CChartObject

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Sınıftan türetilen yöntemler CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Sınıftan türetilen yöntemler CChartObjectEdit

X_Size, Y_Size, BackColor, BackColor, BorderColor, BorderColor, ReadOnly, ReadOnly, TextAlign, TextAlign, Angle, Create

Ayrıca bakınız

Nesne tipleri, Nesne özellikleri, Çizelge köşesi, Nesne bağlama yöntemleri, Grafik nesneleri

State (Get Yöntemi)

"Durum" özelliğinin değerini alır.

```
bool  State() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Durum" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

State (Set Yöntemi)

"Durum" özelliği için yeni değer ayarlar.

```
bool  State(
   bool  state      // yeni durum değeri
   )
```

Parametreler

X

[in] "Durum" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Dosyadan nesne parametreleri yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectButton için OBJ_BUTTON).

CChartObjectSubChart

CChartObjectSubChart sınıfı, "Çizelge" grafik nesnesinin özelliklerine kolay erişim için düzenlenmiş bir sınıftır.

Açıklama

CChartObjectSubChartt sınıfı, "Çizelge" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectSubChart : public CChartObject
```

Başlık

```
#include <ChartObjects\ChartObjectSubChart.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectSubChart

Sınıf Yöntemleri

Oluştur	
Create	"Çizelge" grafik nesnesinin oluşturur
Özellikler	
X_Distance	"X-Uzaklığı" özelliğinin değerini alır/ayarlar
Y_Distance	"Y-Uzaklığı" özelliğinin değerini alır/ayarlar
Corner	"Köşe" özelliğinin değerini alır/ayarlar
X_Size	"X-Büyüklüğü" özelliğinin değerini alır/ayarlar
Y_Size	"Y-Büyüklüğü" özelliğinin değerini alır/ayarlar
Symbol	"Sembol" özelliğinin değerini alır
Period	"Periyot" özelliğinin değerini alır
Scale	"Ölçek" özelliğinin değerini alır/atarlar
DateScale	"Tarih ölçeğini göster" özelliğinin değerini alır
PriceScale	"Fiyat ölçeğini göster" özelliğinin değerini alır
Time	Zaman koordinatındaki değişimler için bir "Saplama" (stub) yöntemi
Price	Fiyat koordinatındaki değişimler için bir "Saplama" yöntemi
Girdi/çıktı	

Oluştur	
virtual Save	Sanal dosya yazım yöntemi
virtual Load	Sanal dosya okuma yöntemi
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Ayrıca bakınız

Nesne tipleri, Nesne özellikleri, Çizelge köşesi, Grafik nesneleri

Create

"Alt-çizelge" (çizelge içinde çizelge) grafik nesnesini oluşturur.

```
bool  Create(
   long     chart_id,      // Çizelge tanımlayıcısı
   string   name,          // Nesne ismi
   int      window,        // Çizelge penceresi
   int      X,             // X koordinatı
   int      Y,             // Y koordinatı
   int      sizeX,         // X büyüklüğü
   int      sizeY          // Y büyüklüğü
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

X

[in] X koordinatı.

Y

[in] Y koordinatı.

sizeX

[in] X büyüklüğü.

sizeY

[in] Y büyüklüğü.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

X_Distance (Get Yöntemi)

"X Uzaklığı" özelliğinin değerini alır.

```
int  X_Distance() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "X Uzaklığı" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

X_Distance (Set Yöntemi)

"X Uzaklığı" özelliğinin değerini ayarlar.

```
bool  X_Distance(
   int  X      // yeni değer
   )
```

Parametreler

X

[in] "X Uzaklığı" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Y_Distance (Get Yöntemi)

"Y Uzaklığı" özelliğinin değerini alır.

```
int  Y_Distance() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Y Uzaklığı" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

Y_Distance (Set Yöntemi)

"Y Uzaklığı" özelliği için yeni değer ayarlar.

```
bool  Y_Distance(
   int  Y      // yeni değer
   )
```

Parametreler

Y

[in] "Y Uzaklığı" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Corner (Get Yöntemi)

"Köşe" özelliğinin değerini alır.

```
ENUM_BASE_CORNER Corner() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Köşe" özelliğinin değeri. Atanmış hiçbir nesne yoksa WRONG_VALUE dönüşü yapar.

Corner (Set Yöntemi)

"Köşe" özelliği için yeni değer ayarlar.

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // yeni değer
   )
```

Parametreler

corner

[in] "Köşe" özelliği için yeni değer

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

X_Size (Get Yöntemi)

"X Büyüklüğü" özelliğinin değerini alır.

```
int  X_Size() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "X Büyüklüğü" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

X_Size (Set Yöntemi)

"X Büyüklüğü" özelliğinin değerini ayarlar.

```
bool  X_Size(
   int  X      // yeni değer
   )
```

Parametreler

X

[in] "X Büyüklüğü" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Y_Size (Get Yöntemi)

"Y Büyüklüğü" özelliğinin değerini alır.

```
int  Y_Size() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Y Büyüklüğü" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

Y_Size (Set Yöntemi)

"Y Büyüklüğü" özelliği için yeni değer ayarlar.

```
bool  Y_Size(
   int  Y      // yeni değer
   )
```

Parametreler

Y

[in] "Y Büyüklüğü" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Symbol (Get Yöntemi)

"Sembol" özelliğinin değerini alır.

```
string  Symbol() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Sembol" özelliğinin değeri. Atanmış bir nesne yoksa "" dönüşü yapar.

Symbol (Set Yöntemi)

"Sembol" özelliği için yeni değer ayarlar.

```
bool  Symbol(
   string  symbol      // yeni sembol
   )
```

Parametreler

symbol

[in] "Sembol" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Period (Get Yöntemi)

"Periyot" özelliğinin değerini alır.

```
int  Period() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Periyot" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

Period (Set Yöntemi)

"Periyot" özelliği için yeni değer ayarlar.

```
bool  Period(
   int  period      // yeni periyot
   )
```

Parametreler

period

[in] "Periyot" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Scale (Get Yöntemi)

"Ölçek" özelliğinin değerini alır.

```
double  Scale() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Ölçek" özelliğinin değeri. Nesne atanmamışsa EMPTY_VALUE değerine dönüş yapar.

Scale (Set Yöntemi)

"Ölçek" özelliği için yeni değer ayarlar.

```
bool  Scale(
   double  scale      // yeni ölçek
   )
```

Parametreler

scale

[in] "Ölçek" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

DateScale (Get Yöntemi)

"Tarih ölçeği" özelliğinin değerini alır.

```
bool  DateScale() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Tarih ölçeği" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

DateScale (Set Yöntemi)

"Tarih ölçeği" özelliğinin değerini ayarlar.

```
bool  DateScale(
   bool  scale      // yeni değer
   )
```

Parametreler

scale

[in] "Tarih ölçeği" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

PriceScale (Get Yöntemi)

"Fiyat ölçeği" özelliğinin değerini alır.

```
bool  PriceScale() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Fiyat ölçeği" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

PriceScale (Set Yöntemi)

"Fiyat ölçeği" özelliği için yeni değer ayarlar.

```
bool  PriceScale(
   bool  scale      // yeni değer
   )
```

Parametreler

scale

[in] "Fiyat ölçeği" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Time

Zaman koordinatının değiştirilmesini engeller.

```
bool  Time(
   datetime  time      // herhangi bir değer
   )
```

Parametreler

time

[in] datetime tipli herhangi bir değer.

Dönüş değeri

Daima 'false'.

Price

Fiyat koordinatının değiştirilmesini engeller.

```
bool  Price(
   double  price      // her hangi bir değer
   )
```

Parametreler

price

[in] double tipli herhangi bir değer.

Dönüş değeri

Daima 'false'.

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Dosyadan nesne parametreleri yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectSubChart için OBJ_CHART).

CChartObjectBitmap

CChartObjectBitmap sınıfı, "Biteşlem" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectBitmap sınıfı, "Biteşlem" grafik nesnesinin özelliklerine kolay erişim sağlar.

Bildirim

```
class CChartObjectBitmap : public CChartObject
```

Başlık

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectBitmap

Sınıf Yöntemleri

Oluştur	
Create	"Biteşlem" grafik nesnesini oluşturur
Özellikler	
BmpFile	"BMP dosya ismi" özelliğini alır/ayarlar
X_Offset	"X-Konumu" özelliğinin değerini alır/ayarlar
Y_Offset	"Y-Konumu" özelliğinin değerini alır/ayarlar
Girdi/çıktı	
virtual Save	Sanal dosya yazım yöntemi
virtual Load	Sanal dosya okuma yöntemi
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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,

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

LevelDescription, GetInteger, GetInteger, SetInteger, SetInteger, GetDouble, GetDouble, SetDouble, SetDouble, GetString, GetString, SetString, SetString, ShiftObject, ShiftPoint

Ayrıca bakınız

Nesne tipleri, Nesne özellikleri, Grafik nesneleri

Create

"Biteşlem" grafik nesnesini oluşturur.

```
bool  Create(
   long      chart_id,      // çizelge tanımlayıcısı
   string    name,          // nesne ismi
   int       window,        // çizelge alt-penceresi
   datetime  time,          // zaman koordinatı
   double    price          // fiyat koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

time

[in] Tutturma noktasının zaman koordinatı.

price

[in] Tutturma noktasının fiyat koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

BmpFile (Get Yöntemi)

Bmp Dosyasının ismini alır.

```
string  BmpFile() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Bmp Dosyası" özelliğinin değeri (dosya ismi). Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

BmpFile (Set Yöntemi)

"Bmp Dosyası" için yeni isim ayarlar.

```
bool  BmpFile(
   string  name      // yeni dosya ismi
   )
```

Parametreler

name

[in] "Bmp Dosyası" için yeni isim değeri.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

X_Offset (Get Yöntemi)

"X-Konumu" özelliğinin değerini alır (grafik nesnesindeki görünür dikdörtgen alanın sol üst köşesinin X koordinatı).

```
int  X_Offset() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "X-Konumu" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

X_Offset (Set Yöntemi)

"X-Konumu" özelliğinin değerini ayarlar (grafik nesnesindeki görünür dikdörtgen alanın sol üst köşesinin X koordinatı). Değer, orjinal resmin sol üst köşesine göre, piksel bazında ayarlanır.

```
bool  X_Offset(
   int  X      // yeni değer
   )
```

Parametreler

X

[in] "X-Konumu" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Y_Offset (Get Yöntemi)

"Y-Konumu" özelliğinin değerini alır (grafik nesnesindeki görünür dikdörtgen alanın sol üst köşesinin Y koordinatı).

```
int  Y_Offset() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Y-Konumu" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

Y_Offset (Set Yöntemi)

"Y-Konumu" özelliğinin değerini ayarlar (grafik nesnesindeki görünür dikdörtgen alanın sol üst köşesinin Y koordinatı). Değer, orjinal resmin sol üst köşesine göre, piksel bazında ayarlanır.

```
bool  Y_Offset(
   int  Y      // yeni değer
   )
```

Parametreler

Y

[in] "Y-Konumu" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Dosyadan nesne parametreleri yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectBitmap için OBJ_BITMAP).

CChartObjectBmpLabel

CChartObjectBmpLabel sınıfı, "Biteşlem Etiketi" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectBmpLabel sınıfı, "Biteşlem Etiketi" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectBmpLabel : public CChartObject
```

Başlık

```
#include <ChartObjects\ChartObjectsBmpControls.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectBmpLabel

Sınıf Yöntemleri

Oluştur	
Create	"Biteşlem Etiketi" grafik nesnesini oluşturur
Özellikler	
X_Distance	"X-Uzaklığı" özelliğinin değerini alır/ayarlar
Y_Distance	"Y-Uzaklığı" özelliğinin değerini alır/ayarlar
X_Offset	"X-Konumu" özelliğinin değerini alır/ayarlar
Y_Offset	"Y-Konumu" özelliğinin değerini alır/ayarlar
Corner	"Köşe" özelliğinin değerini alır/ayarlar
X_Size	"X-Büyüklüğü" özelliğinin değerini alır/ayarlar
Y_Size	"Y-Büyüklüğü" özelliğinin değerini alır/ayarlar
BmpFileOn	Düğme-basılı durum (On) için kullanılacak Bmp Dosyasının ismini alır/ayarlar
BmpFileOff	Düğme-serbest durum (Off) için kullanılacak Bmp Dosyasının ismini alır/ayarlar
State	"Düğme Durumu" (Basılı/Serbest) özelliğinin ismini alır/ayarlar
Time	Zaman koordinatındaki değişimler için bir "Saplama" (stub) yöntemi
Price	Fiyat koordinatındaki değişimler için bir "Saplama" yöntemi

Oluştur	
Girdi/çıktı	
virtual Save	Sanal dosya yazım yöntemi
virtual Load	Sanal dosya okuma yöntemi
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Ayrıca bakınız

Nesne tipleri, Nesne özellikleri, Çizelge köşesi, Grafik nesneleri

Create

"Biteşlem Etiketi" grafik nesnesini oluşturur.

```
bool  Create(
   long     chart_id,      // Çizelge tanımlayıcısı
   string   name,          // Nesne ismi
   int      window,        // Çizelge penceresi
   int      X,             // X koordinatı
   int      Y              // Y koordinatı
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

X

[in] X koordinatı.

Y

[in] Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

X_Distance (Get Yöntemi)

"X Uzaklığı" özelliğinin değerini alır.

```
int  X_Distance() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "X Uzaklığı" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

X_Distance (Set Yöntemi)

"X Uzaklığı" özelliğinin değerini ayarlar.

```
bool  X_Distance(
   int  X      // yeni değer
   )
```

Parametreler

X

[in] "X Uzaklığı" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Y_Distance (Get Yöntemi)

"Y Uzaklığı" özelliğinin değerini alır.

```
int  Y_Distance() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Y Uzaklığı" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

Y_Distance (Set Yöntemi)

"Y Uzaklığı" özelliği için yeni değer ayarlar.

```
bool  Y_Distance(
   int  Y      // yeni değer
   )
```

Parametreler

Y

[in] "Y Uzaklığı" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

X_Offset (Get Yöntemi)

"X-Konumu" özelliğinin değerini alır (grafik nesnesindeki görünür dikdörtgen alanın sol üst köşesinin X koordinatı).

```
int  X_Offset() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "X-Konumu" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

X_Offset (Set Yöntemi)

"X-Konumu" özelliğinin değerini ayarlar (grafik nesnesindeki görünür dikdörtgen alanın sol üst köşesinin X koordinatı). Değer, orjinal resmin sol üst köşesine göre, piksel bazında ayarlanır.

```
bool  X_Offset(
   int  X      // yeni değer
   )
```

Parametreler

X

[in] "X-Konumu" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Y_Offset (Get Yöntemi)

"Y-Konumu" özelliğinin değerini alır (grafik nesnesindeki görünür dikdörtgen alanın sol üst köşesinin Y koordinatı).

```
int  Y_Offset() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "X-Konumu" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

Y_Offset (Set Yöntemi)

"Y-Konumu" özelliğinin değerini ayarlar (grafik nesnesindeki görünür dikdörtgen alanın sol üst köşesinin Y koordinatı). Değer, orjinal resmin sol üst köşesine göre, piksel bazında ayarlanır.

```
bool  Y_Offset(
   int  Y      // yeni değer
   )
```

Parametreler

Y

[in] "Y-Konumu" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Corner (Get Yöntemi)

"Köşe" özelliğinin değerini alır.

```
ENUM_BASE_CORNER Corner() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Köşe" özelliğinin değeri. Atanmış hiçbir nesne yoksa WRONG_VALUE dönüşü yapar.

Corner (Set Yöntemi)

"Köşe" özelliği için yeni değer ayarlar.

```
bool  Corner(
   ENUM_BASE_CORNER  corner      // yeni değer
   )
```

Parametreler

corner

[in] "Köşe" özelliği için yeni değer

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

X_Size

"X Büyüklüğü" özelliğinin değerini alır.

```
int  X_Size() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "X Büyüklüğü" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

Y_Size

"Y Büyüklüğü" özelliğinin değerini alır.

```
int  Y_Size() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Y Büyüklüğü" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

BmpFileOn (Get Yöntemi)

"Bmp düğme-basılı" (ON) özelliği için dosya ismini alır.

```
string  BmpFileOn() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Bmp düğme-basılı" özelliğinin dosya ismi. Atanmış bir nesne yoksa "" dönüşü yapar.

BmpFileOn (Set Yöntemi)

"Bmp düğme-basılı" (ON) özelliği için yeni dosya ismi ayarlar.

```
bool  BmpFileOn(
   string  name      // dosya ismi
   )
```

Parametreler

name

[in] "Bmp düğme-basılı" özelliği için yeni dosya ismi.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

BmpFileOff (Get Yöntemi)

"Bmp düğme-serbest (OFF) özelliği için dosya ismini alır.

```
string  BmpFileOff() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Bmp düğme-serbest" özelliğinin dosya ismi. Atanmış bir nesne yoksa "" dönüşü yapar.

BmpFileOff (Set Yöntemi)

"Bmp düğme-serbest" (OFF) özelliği için yeni dosya ismi ayarlar.

```
bool  BmpFileOff(
   string  name      // dosya ismi
   )
```

Parametreler

name

[in] "Bmp düğme-serbest" özelliği için yeni dosya ismi.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

State (Get Method)

"Durum" özelliğinin değerini alır.

```
bool  State() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Durum" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

State (Set Method)

"Durum" özelliği için yeni değer ayarlar.

```
bool  State(
   bool  state      // yeni durum değeri
   )
```

Parametreler

state

[in] "Durum" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Time

Zaman koordinatının değiştirilmesini engeller.

```
bool  Time(
   datetime  time      // herhangi bir değer
   )
```

Parametreler

time

[in] datetime tipli herhangi bir değer.

Dönüş değeri

Daima 'false'.

Price

Fiyat koordinatının değiştirilmesini engeller.

```
bool  Price(
   double  price      // her hangi bir değer
   )
```

Parametreler

price

[in] double tipli herhangi bir değer.

Dönüş değeri

Daima 'false'.

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Dosyadan nesne parametreleri yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectBmpLabel için OBJ_BITMAP_LABEL).

CChartObjectRectLabel

CChartObjectRectLabel sınıfı, "Dikdörtgen Etiket" grafik nesnesinin özelliklerine kolay erişim sağlamak için geliştirilmiştir.

Açıklama

CChartObjectBmpLabel sınıfı, "Dikdörtgen Etiket" grafik nesnesinin özelliklerine erişim sağlar.

Bildirim

```
class CChartObjectRectLabel : public CChartObjectLabel
```

Başlık

```
#include <ChartObjects\ChartObjectsTxtControls.mqh>
```

Kalıtım hiyerarşisi

CObject
  CChartObject
    CChartObjectText
      CChartObjectLabel
        CChartObjectRectLabel

Sınıf Yöntemleri

Oluştur	
Create	"Dikdörtgen Etiket" grafik nesnesini oluşturur
Özellikler	
X_Size	Yatay büyüklüğü ayarlar
Y_Size	Dikey büyüklüğü ayarlar
BackColor	Arka-plan rengini alır/ayarlar
Angle	Bir saplama (stub) yöntemi.
BorderType	Belirtilen grafiksel nesnenin kenarlık tipini alır/ayarlar
Girdi/çıktı	
virtual Save	Sanal dosya yazım yöntemi
virtual Load	Sanal dosya okuma yöntemi
virtual Type	Sanal tanımlama yöntemi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CChartObjectText

Angle, Angle, Font, Font, FontSize, FontSize, Anchor, Anchor, Create

Sınıftan türetilen yöntemler CChartObjectLabel

X_Distance, X_Distance, Y_Distance, Y_Distance, X_Size, Y_Size, Corner, Corner, Time, Price, Create

Ayrıca bakınız

Nesne tipleri, Nesne özellikleri, Grafik nesneleri

Create

"CChartObjectRectLabel" grafik nesnesini oluşturur.

```
bool  Create(
   long     chart_id,      // Çizelge tanımlayıcısı
   string   name,          // Nesne ismi
   int      window,        // Çizelge alt-penceresi
   int      X,             // X koordinatı
   int      Y,             // Y koordinatı
   int      sizeX,         // Yatay büyüklük
   int      sizeY          // Dikey büyüklük
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı (0 değeri mevcut çizelgeyi temsil eder).

name

[in] Oluşturulacak nesnenin benzersiz ismi.

window

[in] Çizelge alt-pencere numarası (0 değeri ana pencereyi temsi eder).

X

[in] X koordinatı.

Y

[in] Y koordinatı.

sizeX

[in] Yatay büyüklük.

sizeY

[in] Dikey büyüklük.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

X_Size

"X Büyüklüğü" özelliğinin değerini ayarlar.

```
bool  X_Size(
   int  size      // Yatay büyüklük
   )
```

Parametreler

size

[in] Yeni yatay büyüklük değeri.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Not

"X Büyüklüğü" ve "Y Büyüklüğü" özellik değerlerini almak için CChartObjectLabel ebeveyn sınıfının X_Size ve Y_Size yöntemlerini kullanın.

Y_Size

"Y Büyüklüğü" özelliğinin değerini ayarlar.

```
bool  Y_Size(
   int  size      // Dikey büyüklük
   )
```

Parametreler

size

[in] Yeni dikey büyüklük değeri.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Not

"X Büyüklüğü" ve "Y Büyüklüğü" özellik değerlerini almak için CChartObjectLabel ebeveyn sınıfının X_Size ve Y_Size yöntemlerini kullanın.

BackColor

Arka-plan rengini alır.

```
color  BackColor() const
```

Dönüş değeri

Sınıf örneğine atanmış grafik nesnesinin arka-plan rengi. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

BackColor

Arka-plan rengini ayarlar.

```
bool  BackColor(
   color  new_color      // yeni renk
   )
```

Parametreler

new_color

[in] Yeni arka-plan rengi.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Angle

Açı özelliği için bir saplama (stub) yöntemi.

```
bool  Angle(
   double  angle      // herhangi bir değer
   )
```

Parametreler

angle

[in] double tipli herhangi bir değer.

Dönüş değeri

Daima 'false'.

BorderType

Kenarlık tipini alır.

```
int  BorderType() const
```

Dönüş değeri

Sınıf örneğine atanmış grafik nesnesinin kenarlık tipi. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

BorderType

Kenarlık tipini ayarlar.

```
bool  BorderType(
   int  type      // kenarlık tipi
   )
```

Parametreler

type

[in] Yeni kenarlık tipi.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Dosyadan nesne parametreleri yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen fonksiyonu ile açılmış olan ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChartObjectRectangleLabel için OBJ_RECTANGLE_LABEL).

Özel grafikler

Bu bölüm kullanıcı tanımlı grafiklerle çalışmak için düzenlenmiştir.

Bunlar, özel çizelgeler yapmak, boyamalar ve veri görselleştirme için kullanılabilir.

Grafiksel nesnelerin ve temel şekillerin oluşturulması, pay grafiklerin ve eğrilerin çizilmesi bu bölümde yer alan sınıfların işlevlerindendir. Sınıflarda, çeşitli nesne görselleştirme seçenekleri sunulmuştur: çizgi renginin ve stilinin değişimi, dolgu, çizelge üzerinde verilerle çalışma, vb.

Sınıf	Açıklama
CCanvas	Kullanıcı tanımlı çizimleri kolayca oluşturmak için düzenlenmiş bir sınıftır
CChartCanvas	Çizelge ve çizelge bileşenleri oluşturmak için tasarlanmış temel sınıf
CHistogramChart	Histogram çizimi için tasarlanmış olan sınıf
CLineChart	Eğri çizimi için tasarlanmış olan sınıf
CPieChart	Pay grafiklerin çizimi için tasarlanmış sınıf

CCanvas

CCanvas, kullanıcı tanımlı imgeler oluşturmak için düzenlenmiş bir sınıftır.

Açıklama

CCanvas, grafiksel kaynağın — bir çizelge nesnesine bağlanarak veya bağlanmadan — oluşturulmasını ve basit grafiklerin çizilmesini sağlar.

Bildirim

```
class CCanvas
```

Başlık

```
#include <Canvas\Canvas.mqh>
```

Kalıtım hiyerarşisi

CCanvas

İlk nesil

CChartCanvas, CFlameCanvas

Gruplarına göre sınıf yöntemleri

Oluşturma	
Attach	OBJ_BITMAP_LABEL nesnesini CCanvas sınıfına tutturur
Create	Herhangi bir çizelge nesnesine bağlamadan bir grafiksel kaynak oluşturur
CreateBitmap	Bir çizelge nesnesine tutturulmuş grafiksel kaynak oluşturur
CreateBitmapLabel	Bir çizelge nesnesine tutturulmuş grafiksel kaynak oluşturur
Destroy	Grafiksel kaynağı yok eder
Özellikler	
ChartObjectName	Tuturulmuş çizelge nesnesinin ismini alır
ResourceName	Grafiksel kaynağın ismini alır
Width	Bir grafiksel kaynağın genişlik bilgisini alır
Height	Bir grafiksel kaynağın yükseklik bilgisini alır
LineStyleSet	Çizgi stilini ayarlar
Ekrandaki bir nesneyi günceller	
Update	Değişiklikleri ekranda gösterir

Oluşturma	
Resize	Bir grafiksel kaynağı yeniden boyutlandırır
Bir renk ile doldurma veya silme	
Erase	Belirtilen renk ile doldurur veya siler
Veri erişimi	
PixelGet	Belirtilen koordinatlardaki noktanın rengini alır
PixelSet	Belirtilen koordinatlardaki noktanın rengini ayarlar
Basit çizimler	
LineVertical	Dikey bir çizgi çizer
LineHorizontal	Yatay bir çizgi çizer
Line	Serbest çizgi çizer
Polyline	Devamlı (çoklu) çizgi çizer
Polygon	Bir çokgen çizer
Rectangle	Dikdörtgen çizer
Circle	Bir daire çizer
Triangle	Bir üçgen çizer
Ellipse	Bir elips çizer
Arc	Bir elips yayı çizer
Pie	Bir elips parçası çizer
Dolgulu basit çizimler	
FillRectangle	Doldurulmuş bir dikdörtgen çizer
FillCircle	Doldurulmuş bir daire çizer
FillTriangle	Doldurulmuş bir üçgen çizer
FillPolygon	Dolgulu poligon çizer
FillEllipse	Dolgulu elips çizer
Fill	Belirlenen alanı doldurur
Antialiasing kullanılan basit çizimler	
PixelSetAA	Bir piksel çizer
LineAA	Bir çizgi çizer
PolylineAA	Devamlı (çoklu) çizgi çizer

Oluşturma	
PolygonAA	Bir çokgen çizer
TriangleAA	Bir üçgen çizer
CircleAA	Bir daire çizer
EllipseAA	Bir elips çizer
LineWu	Bir çizgi çizer
PolylineWu	Devamlı (çoklu) çizgi çizer
PolygonWu	Bir çokgen çizer
TriangleWu	Bir üçgen çizer
CircleWu	Bir daire çizer
EllipseWu	Bir elips çizer
LineThick	Antialiasing algoritması kullanarak belirtilen kalınlıkta bir serbest-çizgi parçası çizer.
LineThickVertical	Antialiasing algoritması kullanarak belirtilen kalınlıkta bir dikey serbest-çizgi parçası çizer.
LineThickHorizontal	Antialiasing algoritması kullanarak belirtilen kalınlıkta bir yatay serbest-çizgi parçası çizer.
PolygonSmooth	İki antialiasing algoritması kullanarak belirtilen kalınlıkta bir çokgen çizer.
PolygonThick	Antialiasing algoritması kullanarak belirtilen kalınlıkta bir çokgen çizer.
PolylineSmooth	İki antialiasing algoritması kullanarak belirtilen kalınlıkta bir çoklu çizgi çizer.
PolylineThick	Antialiasing algoritması kullanarak belirtilen kalınlıkta bir çoklu çizgi çizer.
Metin	
FontSet	Yazı-tipi parametrelerini düzenler
FontNameSet	Yazı-tipi ismini ayarlar
FontSizeSet	Yazı-tipi boyutunu ayarlar
FontFlagsSet	Yazı-tipi bayraklarını ayarlar
FontAngleSet	Yazı-tipi eğim açısını ayarlar
FontGet	Yazı-tipi parametrelerini alır
FontNameGet	Yazı-tipi ismini alır
FontSizeGet	Yazı-tipi boyutunu alır

Oluşturma	
FontFlagsGet	Yazı-tipi bayraklarını ayarlar
FontAngleGet	Yazı-tipi eğim açısını alır
TextOut	Metni görüntüler
TextWidth	Metin genişliği bilgisini alır
TextHeight	Metin yüksekliği bilgisini alır
TextSize	Metin boyutunu alır
Saydamlık	
TransparentLevelSet	Saydamlık seviyesini ayarlar
Girdi/çıktı	
LoadFromFile	Bir BMP dosyasından resim okur

Attach

Bir OBJ_BITMAP_LABEL nesnesinden grafiksel kaynak alır ve bunu bir CCanvas sınıf örneğine iliştirir.

```
bool  Attach(
   const long          chart_id,                          // çizelge tanıtıcısı
   const string        objname,                           // nesne ismi
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // renk işleme yöntemi
  );
```

Bir OBJ_BITMAP_LABEL nesnesi için grafiksel kaynak oluşturur ve bunu bir CCanvas sınıf örneğine iliştirir.

```
bool  Attach(
   const long          chart_id,                          // çizelge tanıtıcısı
   const string        objname,                           // nesne ismi
   const int           width,                             // piksel cinsinden resir
   const int           height,                            // piksel cinsinden resir
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // renk işleme yöntemi
  );
```

Parametreler

chart_id

[out] Çizelge tanımlayıcısı.

objname

[in] Grafiksel nesne ismi.

width

[in] Kaynaktaki resmin genişliği.

height

[in] Kaynaktaki resmin yüksekliği.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] alfa kanalının işlenme yöntemi. Alfa kanalı varsayılan olarak gözardı edilir.

Dönüş Değeri

Başarılı ise 'true', nesne iliştirilemediyse 'false'.

Arc

(x1,y1) ve (x2,y2) köşe koordinatlarıyla tanımlanan bir dikdörtgen içinde bir elips yayı çizer. Yay sınırları elips merkezinden gelen çizgilerle - (x3,y3) ve (x4,y4) koordinatlarıyla - kesilir.

```
void  Arc(
   int          x1,      // dikdörtgenin sol üst köşesinin X koordinatı
   int          y1,      // dikdörtgenin sol üst köşesinin Y koordinatı
   int          x2,      // dikdörtgenin ssağ alt köşesinin X koordinatı
   int          y2,      // dikdörtgenin ssağ alt köşesinin Y koordinatı
   int          x3,      // yay sınırını belirleyen ilk noktanın X koordinatı
   int          y3,      // yay sınırını belirleyen ilk noktanın Y koordinatı
   int          x4,      // yay sınırını belirleyen ikinci noktanın X koordinatı
   int          y4,      // yay sınırını belirleyen ikinci noktanın Y koordinatı
   const uint   clr      // renk
   );
```

Parametreler

x1

[in] Dikdörtgenin sol üst köşesinin X koordinatı.

y1

[in] Dikdörtgenin sol üst köşesinin Y koordinatı.

x2

[in] Dikdörtgenin sağ alt köşesinin X koordinatı.

y2

[in] Dikdörtgenin sağ alt köşesinin Y koordinatı.

x3

[in] Dikdörtgen merkezinden gelip yay sınırını belirleyen çizgi için, ilk noktanın X koordinatı.

y3

[in] Dikdörtgen merkezinden gelip yay sınırını belirleyen çizgi için, ilk noktanın Y koordinatı.

x4

[in] Dikdörtgen merkezinden gelip yay sınırını belirleyen çizgi için, ikinci noktanın X koordinatı.

y4

[in] Dikdörtgen merkezinden gelip yay sınırını belirleyen çizgi için, ikinci noktanın Y koordinatı.

clr

[in] ARGB biçimli renk. Rengi argb biçimine dönüştürmek için ColorToARGB() fonksiyonunu kullanın.

rx ve ry uzunlukları ile tanımlı dikdörtgen içinde, (x,y) merkezli elipsin bir yayını çizer. Yay sınırları elips merkezinden gelen fi3 ve fi4 açılarına sahip ışınlarla kesilir.

```
void  Arc(
```

```
   int          x,        // elips merkezinin X koordinatı
   int          y,        // elips merkezinin Y koordinatı
   int          rx,       // elipsin yatay (X ekseni)yarıçapı
   int          ry,       // elipsin dikey (Y ekseni)yarıçapı
   int          fi3,      // elips yayının ilk noktasının açısı
   int          fi4,      // elips yayının ikinci noktasının açısı
   const uint   clr       // renk
   );
```

rx ve ry uzunlukları ile tanımlı dikdörtgen içinde, (x,y) merkezli elipsin bir yayını çizer ve yay sınırlarının koordinatlarına dönüş yapar. Yay sınırları elips merkezinden gelen fi3 ve fi4 açılarına sahip ışınlarla kesilir.

```
void  Arc(
   int          x,        // elips merkezinin X koordinatı
   int          y,        // elips merkezinin Y koordinatı
   int          rx,       // elipsin yatay (X ekseni)yarıçapı
   int          ry,       // elipsin dikey (Y ekseni)yarıçapı
   int          fi3,      // elips yayının ilk noktasının açısı
   int          fi4,      // elips yayının ikinci noktasının açısı
   int&         x3,       // yayın ilk sınırının X koordinatı
   int&         y3,       // yayın ilk sınırının Y koordinatı
   int&         x4,       // yayın ikinci sınırının X koordinatı
   int&         y4,       // yayın ikinci sınırının Y koordinatı
   const uint   clr       // renk
   );
```

Parametreler

x

[in] Elips merkezinin X koordinatı.

y

[in] Elips merkezinin Y koordinatı.

rx

[in] Elipsin X eksenindeki yarı çapı (piksel cinsinden).

ry

[in] Elipsin Y eksenindeki yarı çapı (piksel cinsinden).

fi3

[in] İlk yay sınırını belirten açının radyan cinsinden değeri.

fi4

[in] İkinci yay sınırını belirten açının radyan cinsinden değeri.

x3

[out] İlk yay sınırının X koordinatını alacak değişken.

y3

[out] İlk yay sınırının Y koordinatını alacak değişken.

x4

[out] İkinci yay sınırının X koordinatını alacak değişken.

y4

[out] İkinci yay sınırının Y koordinatını alacak değişken.

clr

[in] ARGB biçimli renk. Rengi argb biçimine dönüştürmek için ColorToARGB() fonksiyonunu kullanın.

Sınıf yöntemlerinin çağrılmasına örnek:

```
#include <Canvas\Canvas.mqh>
CCanvas canvas;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int      Width=600;
   int      Height=400;
//--- kanvası oluştur
   if(!canvas.CreateBitmapLabel(0,0,"CirclesCanvas",30,30,Width,Height))
     {
      Print("Hata, kanvas oluşturulamadı: ",GetLastError());
     }
//--- kanvası temizle
   canvas.Erase(clrWhite);
//--- dikdörtgen çiz
   canvas.Rectangle(215-190,215-120,215+190,215+120,clrGray);
//--- ilk yayı çiz
   canvas.Arc(215,215, 190,120,M_PI_4,2*M_PI-M_PI_4,ColorToARGB(clrRed));
   int x1,y1,x2,y2;
//--- ikinci yayı çiz
   canvas.Arc(215,215, 190,120,2*M_PI-M_PI_4,2*M_PI+M_PI_4,x1,y1,x2,y2,ColorToARGB(cl
//--- yay koordinatlarını yazdır
   PrintFormat("Yayın ilk noktası: (%G,%G), yayın ikinci noktası (%G,%G)",x1,y1,x2,y2)
   canvas.CircleAA(x1,y1,3, ColorToARGB(clrRed));
   canvas.CircleAA(x2,y2,3, ColorToARGB(clrBlue));
//--- güncellenen kanvası görüntüle
   canvas.Update();
  }
```

Pie

(x1,y1) ve (x2,y2) köşe koordinatlarıyla tanımlanan bir dikdörtgen içinde bir dolguulu elips dilimi çizer. Dilim yayının sınırları elips merkezinden gelen çizgilerle - (x3,y3) ve (x4,y4) koordinatlarıyla - kesilir.

```
void  Pie(
   int          x1,      // dikdörtgenin sol üst köşesinin X koordinatı
   int          y1,      // dikdörtgenin sol üst köşesinin Y koordinatı
   int          x2,      // dikdörtgenin ssağ alt köşesinin X koordinatı
   int          y2,      // dikdörtgenin ssağ alt köşesinin Y koordinatı
   int          x3,      // yay sınırını belirleyen ilk noktanın X koordinatı
   int          y3,      // yay sınırını belirleyen ilk noktanın Y koordinatı
   int          x4,      // yay sınırını belirleyen ikinci noktanın X koordinatı
   int          y4,      // yay sınırını belirleyen ikinci noktanın Y koordinatı
   const uint   clr,     // çizgi rengi
   const uint   fill_clr // dolgu rengi
   );
```

Parametreler

x1

[in] Dikdörtgenin sol üst köşesinin X koordinatı.

y1

[in] Dikdörtgenin sol üst köşesinin Y koordinatı.

x2

[in] Dikdörtgenin sağ alt köşesinin X koordinatı.

y2

[in] Dikdörtgenin sağ alt köşesinin Y koordinatı.

x3

[in] Dikdörtgen merkezinden gelip yay sınırını belirleyen çizgi için, ilk noktanın X koordinatı.

y3

[in] Dikdörtgen merkezinden gelip yay sınırını belirleyen çizgi için, ilk noktanın Y koordinatı.

x4

[in] Dikdörtgen merkezinden gelip yay sınırını belirleyen çizgi için, ikinci noktanın X koordinatı.

y4

[in] Dikdörtgen merkezinden gelip yay sınırını belirleyen çizgi için, ikinci noktanın Y koordinatı.

clr

[in] ARGB biçiminde, şekil sınırlarının rengi.

fill_clr

[in] ARGB biçiminde, dolgu rengi. Rengi argb biçimine dönüştürmek için ColorToARGB() fonksiyonunu kullanın.

rx ve ry uzunlukları ile tanımlı dikdörtgen içinde, (x,y) merkezli elipse göre bir dolgulu dilim çizer. Dilim yayının sınırları elips merkezinden gelen fi3 ve fi4 açılarına sahip ışınlarla kesilir.

```
void  Pie(
   int          x,        // elips merkezinin X koordinatı
   int          y,        // elips merkezinin Y koordinatı
   int          rx,       // elipsin yatay (X ekseni)yarıçapı
   int          ry,       // elipsin dikey (Y ekseni)yarıçapı
   int          fi3,      // elips yayının ilk noktasının açısı
   int          fi4,      // elips yayının ikinci noktasının açısı
   const uint   clr,      // çizgi rengi
   const uint   fill_clr // dolgu rengi
   );
```

rx ve ry uzunlukları ile tanımlı dikdörtgen içinde, (x,y) merkezli elips için bir dolgulu dilim çizer ve yay sınırlarının koordinatlarına dönüş yapar. Dilim yayının sınırları elips merkezinden gelen fi3 ve fi4 açılarına sahip ışınlarla kesilir.

```
void  Pie(
   int          x,        // elips merkezinin X koordinatı
   int          y,        // elips merkezinin Y koordinatı
   int          rx,       // elipsin yatay (X ekseni)yarıçapı
   int          ry,       // elipsin dikey (Y ekseni)yarıçapı
   int          fi3,      // elips yayının ilk noktasının açısı
   int          fi4,      // elips yayının ikinci noktasının açısı
   int&         x3,       // yayın ilk sınırının X koordinatı
   int&         y3,       // yayın ilk sınırının Y koordinatı
   int&         x4,       // yayın ikinci sınırının X koordinatı
   int&         y4,       // yayın ikinci sınırının Y koordinatı
   const uint   clr,      // çizgi rengi
   const uint   fill_clr // dolgu rengi
   );
```

Parametreler

x

[in] Elips merkezinin X koordinatı.

y

[in] Elips merkezinin Y koordinatı.

rx

[in] Elipsin X eksenindeki yarı çapı (piksel cinsinden).

ry

[in] Elipsin X eksenindeki yarı çapı (piksel cinsinden).

fi3

[in] İlk yay sınırını belirten açının radyan cinsinden değeri.

fi4

[in] İkinci yay sınırını belirten açının radyan cinsinden değeri.

x3

[out] İlk yay sınırının X koordinatını alacak değişken.

y3

[out] İlk yay sınırının Y koordinatını alacak değişken.

x4

[out] İkinci yay sınırının X koordinatını alacak değişken.

y4

[out] İkinci yay sınırının Y koordinatını alacak değişken.

clr

[in] ARGB biçiminde, şekil sınırlarının rengi.

fill_clr

[in] ARGB biçiminde, dolgu rengi. Rengi argb biçimine dönüştürmek için ColorToARGB() fonksiyonunu kullanın.

Sınıf yöntemlerinin çağrılmasına örnek:

```
#include <Canvas\Canvas.mqh>
CCanvas canvas;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   int      Width=600;
   int      Height=400;
//--- kanvası oluştur
   if(!canvas.CreateBitmapLabel(0,0,"CirclesCanvas",30,30,Width,Height))
     {
      Print("Hata, kanvas oluşturulamadı: ",GetLastError());
     }
//--- kanvası temizle
   canvas.Erase(clrWhite);
//--- dikdörtgen çiz
   canvas.Rectangle(215-190,215-120,215+190,215+120,clrGray);
//--- ilk dilimi içiz
   canvas.Pie(215,215, 190,120,M_PI_4,2*M_PI-M_PI_4,ColorToARGB(clrBlue),ColorToARGB(c
//--- ikinci dilimi çiz
   canvas.Pie(215,215, 190,120,2*M_PI-M_PI_4,2*M_PI+M_PI_4,ColorToARGB(clrGreen),Color
//--- güncellenen kanvası görüntüle
   canvas.Update();
   DebugBreak();
  }
```

FillPolygon

Dolgulu poligon çizer.

```
void  FillPolygon(
   int&          x,       // poligonun X koordinatlarını içeren dizi
   int&          y,       // poligonun Y koordinatlarını içeren dizi
   const uint    clr      // renk
   );
```

Parametreler

x

[in] Poligonun X koordinatlarını içeren dizi.

y

[in] Poligonun Y koordinatlarını içeren dizi.

clr

[in] ARGB biçiminde renk.

FillEllipse

Belirtilen koordinatlarla tanımlu bir dikdörgen içinde dolgulu bir elips çizer.

```
void  FillPolygon(
   int          x1,      // dikdörtgenin sol üst köşesinin X koordinatı
   int          y1,      // dikdörtgenin sol üst köşesinin Y koordinatı
   int          x2,      // dikdörtgenin ssağ alt köşesinin X koordinatı
   int          y2,      // dikdörtgenin ssağ alt köşesinin Y koordinatı
   const uint   clr       // elips rengi
   );
```

Parametreler

x1

[in] Dikdörtgenin sol üst köşesinin X koordinatı.

y1

[in] Dikdörtgenin sol üst köşesinin Y koordinatı.

x2

[in] Dikdörtgenin sağ alt köşesinin X koordinatı.

y2

[in] Dikdörtgenin sağ alt köşesinin Y koordinatı.

clr

[in] ARGB biçiminde renk.

GetDefaultColor

İndisine göre bir ön-tanımlı renge dönüş yapar.

```
static uint  GetDefaultColor(
   const uint  i      // indis
   );
```

Parametreler

i

[in] rengi almak için kullanılacak indis.

Dönüş Değeri

Renk.

ChartObjectName

Bağlanmış bir çizelge nesnesinin ismini alır.

```
string  ChartObjectName();
```

Dönüş değeri

Bağlanmış çizelge nesnesinin ismi

Circle

Bir daire çizer

```
void  Circle(
   int          x,        // X koordinatı
   int          y,        // Y koordinatı
   int          r,        // yarıçap
   const uint   clr       // renk
   );
```

Parametreler

x

[in] Daire merkezinin X koordinatı.

y

[in] Daire merkezinin Y koordinatı.

r

[in] Daire yarıçapı.

clr

[in] ARGB biçiminde renk bilgisi.

CircleAA

Antialiasing algoritması kullanarak bir daire çizer

```
void  CircleAA(
   const int      x,        // X koordinatı
   const int      y,        // Y koordinatı
   const double   r,        // yarıçap
   const uint     clr       // renk
   );
```

Parametreler

x

[in] Daire merkezinin X koordinatı.

y

[in] Daire merkezinin Y koordinatı.

r

[in] Daire yarıçapı.

clr

[in] ARGB biçiminde renk bilgisi.

CircleWu

Wu antialiasing algoritması kullanarak bir daire çizer.

```
void  CircleWu(
   const int      x,        // X koordinatı
   const int      y,        // Y koordinatı
   const double   r,        // yarıçap
   const uint     clr       // renk
   );
```

Parametreler

x

[in] Daire merkezinin X koordinatı.

y

[in] Daire merkezinin Y koordinatı.

r

[in] Daire yarıçapı.

clr

[in] ARGB biçiminde renk bilgisi.

Create

Herhangi bir çizelge nesnesine bağlamadan bir grafiksel kaynak oluşturur.

```
virtual bool  Create(
   const string       name,                              // isim
   const int          width,                             // genişlik
   const int          height,                            // yükseklik
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // biçim
   );
```

Parametreler

name

[in] Grafiksel kaynak ismi için bir kök. Kaynak ismi, oluşturma işlemi sırasında kök kelimeye pseudo-rassal bir dizgi eklenerek meydana getirilir.

width

[in] Piksel bazında genişlik (X-ekseni üzerindeki boyut).

height

[in] Piksel bazında yükseklik (Y-ekseni üzerindeki boyut).

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Renk işleme yöntemi. Renk işleme yöntemleri hakkında daha fazla bilgi için ResourceCreate() fonksiyonunun açıklamasına bakın.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'

CreateBitmap

Belli bir çizelge nesnesine bağlanan bir grafiksel kaynak oluşturur.

1. Mevcut çizelgenin ana penceresinde bir grafiksel kaynak oluşturur.

```
bool  CreateBitmap(
   const string       name,                                  // isim
   const datetime     time,                                  // zaman
   const double       price,                                 // fiyat
   const int          width,                                 // genişlik
   const int          height,                                // yükseklik
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // biçim
   );
```

2. Çizelge tanımlayıcısı ve alt-pencere numarası ile bir grafiksel nesne oluşturur.

```
bool  CreateBitmap(
   const long         chart_id,                              // çizelge tanımlayıcısı
   const int          subwin,                                // alt-pencere numarası
   const string       name,                                  // isim
   const datetime     time,                                  // zaman
   const double       price,                                 // fiyat
   const int          width,                                 // genişlik
   const int          height,                                // yükseklik
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA       // biçim
   );
```

Parametreler

chart_id

[in] Nesnenin oluşturulacağı çizelgenin tanımlayıcısı.

subwin

[in] Nesnenin oluşturulacağı alt-pencerenin numarası.

name

[in] Çizelge nesnesinin ismi ve grafiksel kaynak ismi için bir kök.

time

[in] Çizelge nesnesi tutturma noktasının zaman koordinatı.

price

[in] Çizelge nesnesi tutturma noktasının fiyat koordinatı.

width

[in] Grafik nesnesinin piksel bazında genişliği (X-ekseni üzerindeki boyut).

height

[in] Grafik nesnesinin piksel bazında yüksekliği (Y-ekseni üzerindeki boyut).

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Renk işleme yöntemi. Renk işleme yöntemleri hakkında daha fazla bilgi için ResourceCreate() fonksiyonunun açıklamasına bakın.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'

Not

Fonksiyonun ilk versiyonunun kullanılması durumunda, nesne mevcut çizelgenin ana penceresinde oluşturulur.

Nesnenin boyutu grafiksel kaynağın boyutuyla örtüşür.

CreateBitmapLabel

Belli bir çizelge nesnesine bağlanan bir grafiksel kaynak oluşturur.

1. Mevcut çizelgenin ana penceresinde bir grafiksel kaynak oluşturur.

```
bool  CreateBitmapLabel(
   const string       name,                              // isim
   const int          x,                                 // X koordinatı
   const int          y,                                 // Y koordinatı
   const int          width,                             // genişlik
   const int          height,                            // yükseklik
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // biçim
   );
```

2. Çizelge tanımlayıcısı ve alt-pencere numarası ile bir grafiksel nesne oluşturur.

```
bool  CreateBitmapLabel(
   const long         chart_id,                          // çizelge tanımlayıcısı
   const int          subwin,                            // alt-pencere numarası
   const string       name,                              // isim
   const int          x,                                 // X koordinatı
   const int          y,                                 // Y koordinatı
   const int          width,                             // genişlik
   const int          height,                            // yükseklik
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // biçim
   );
```

Parametreler

chart_id

[in] Nesnenin oluşturulacağı çizelgenin tanımlayıcısı.

subwin

[in] Nesnenin oluşturulacağı alt-pencerenin numarası.

name

[in] Çizelge nesnesinin ismi ve grafiksel kaynak ismi için bir kök.

x

[in] Çizelge nesnesi tutturma noktasının X koordinatı.

y

[in] Çizelge nesnesi tutturma noktasının Y koordinatı.

width

[in] Grafik nesnesinin piksel bazında genişliği (X-ekseni üzerindeki boyut).

height

[in] Grafik nesnesinin piksel bazında yüksekliği (Y-ekseni üzerindeki boyut).

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Renk işleme yöntemi. Renk işleme yöntemleri hakkında daha fazla bilgi için ResourceCreate() fonksiyonunun açıklamasına bakın.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'

Not

Fonksiyonun ilk versiyonunun kullanılması durumunda, nesne mevcut çizelgenin ana penceresinde oluşturulur.

Nesnenin boyutu grafiksel kaynağın boyutuyla örtüşür.

Destroy

Bir grafiksel kaynağı yok eder.

```
void  Destroy();
```

Not

Grafiksel kaynağın bir çizelge nesnesine bağlı olması durumunda, sonuncu olan silinir.

Ellipse

İki nokta kullanarak bir elips çizer.

```
void  Ellipse(
   int          x1,      // X koordinatı
   int          y1,      // Y koordinatı
   int          x2,      // X koordinatı
   int          y2,      // Y koordinatı
   const uint   clr      // renk
   );
```

Parametreler

x1

[in] Birinci elips şekillendirme noktasının X koordinatı.

y1

[in] Birinci elips şekillendirme noktasının Y koordinatı.

x2

[in] İkinci elips şekillendirme noktasının X koordinatı.

y2

[in] İkinci elips şekillendirme noktasının Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

EllipseAA

Antialiasing algoritması, İki nokta kullanarak bir elips çizer

```
void  EllipseAA(
   int          x1,      // X koordinatı
   int          y1,      // Y koordinatı
   int          x2,      // X koordinatı
   int          y2,      // Y koordinatı
   const uint   clr      // renk
   );
```

Parametreler

x1

[in] Birinci elips şekillendirme noktasının X koordinatı.

y1

[in] Birinci elips şekillendirme noktasının Y koordinatı.

x2

[in] İkinci elips şekillendirme noktasının X koordinatı.

y2

[in] İkinci elips şekillendirme noktasının Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

EllipseWu

Wu antialiasing algoritması, İki nokta kullanarak bir elips çizer.

```
void  EllipseWu(
   int          x1,      // X koordinatı
   int          y1,      // Y koordinatı
   int          x2,      // X koordinatı
   int          y2,      // Y koordinatı
   const uint   clr      // renk
   );
```

Parametreler

x1

[in] Birinci elips şekillendirme noktasının X koordinatı.

y1

[in] Birinci elips şekillendirme noktasının Y koordinatı.

x2

[in] İkinci elips şekillendirme noktasının X koordinatı.

y2

[in] İkinci elips şekillendirme noktasının Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

Erase

Belirtilen renk ile doldurur veya siler.

```
void  Erase(
   const uint  clr=0      // renk
   );
```

Parametreler

clr=0

[in] ARGB biçiminde renk bilgisi.

Fill

Bir alanı doldurur.

```
void  Fill(
   int          x,        // X koordinatı
   int          y,        // Y koordinatı
   const uint   clr       // renk
   );
```

Parametreler

x

[in] Doldurma başlangıç noktasının X koordinatı.

y

[in] Doldurma başlangıç noktasının Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

FillCircle

Doldurulmuş bir daire çizer.

```
void  FillCircle(
   int          x,        // X koordinatı
   int          y,        // Y koordinatı
   int          r,        // yarıçap
   const uint   clr       // renk
   );
```

Parametreler

x

[in] Doldurulmuş dairenin merkezinin X koordinatı.

y

[in] Doldurulmuş dairenin merkezinin Y koordinatı.

r

[in] Doldurulmuş dairenin yarıçapı.

clr

[in] ARGB biçiminde renk bilgisi.

FillRectangle

Doldurulmuş bir dikdörtgen çizer.

```
void  FillRectangle(
   int          x1,      // X koordinatı
   int          y1,      // Y koordinatı
   int          x2,      // X koordinatı
   int          y2,      // Y koordinatı
   const uint   clr      // renk
   );
```

Parametreler

x1

[in] Birinci dikdörtgen şekillendirme noktasının X koordinatı.

y1

[in] Birinci dikdörtgen şekillendirme noktasının Y koordinatı.

x2

[in] İkinci dikdörtgen şekillendirme noktasının X koordinatı.

y2

[in] İkinci dikdörtgen şekillendirme noktasının Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

FillTriangle

Doldurulmuş bir üçgen çizer.

```
void  FillTriangle(
   int          x1,       // X koordinatı
   int          y1,       // Y koordinatı
   int          x2,       // X koordinatı
   int          y2,       // Y koordinatı
   int          x3,       // X koordinatı
   int          y3,       // Y koordinatı
   const uint   clr       // renk
   );
```

Parametreler

x1

[in] Üçgenin ilk köşesinin X koordinatı.

y1

[in] Üçgenin ilk köşesinin Y koordinatı.

x2

[in] Üçgenin ikinci köşesinin X koordinatı.

y2

[in] Üçgenin ikinci köşesinin Y koordinatı.

x3

[in] Üçgenin üçüncü köşesinin X koordinatı.

y3

[in] Üçgenin üçüncü köşesinin Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

FontAngleGet

Yazı-tipi eğim açısını alır.

```
uint  FontAngleGet();
```

Dönüş değeri

yazı-tipi eğim açısı

FontAngleSet

Yazı-tipi eğim açısını ayarlar.

```
bool  FontAngleSet(
   uint  angle      // açı
   );
```

Parametreler

angle

[in] Ondalık değerler ile yazı-tipi eğim açısı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'

FontFlagsGet

Yazı-tipi bayraklarını alır.

```
uint  FontFlagsGet();
```

Dönüş değeri

Yazı-tipi bayrakları

FontFlagsSet

Yazı-tipi bayraklarını ayarlar.

```
bool  FontFlagsSet(
   uint  flags      // bayraklar
   );
```

Parametreler

flags

[in] Yazı-tipi oluşturma bayrakları. Bayraklar hakkında daha fazla bilgi için TextSetFont() fonksiyonunun açıklamasına bakın.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'

FontGet

Mevcut yazı-tipi parametrelerini alır.

```
void  FontGet(
   string&  name,       // isim
   int&     size,       // boyut
   uint&    flags,      // bayraklar
   uint&    angle       // eğim açısı
   );
```

Parametreler

name

[out] Yazı-tipi ismine dönüş yapmak için kullanılan değişkenin referansı.

size

[out] Yazı-tipi boyutuna dönüş yapmak için kullanılan değişkenin referansı.

flags

[out] Yazı-tipi bayraklarına dönüş yapmak için kullanılan değişkenin referansı.

angle

[out] Yazı-tipinin eğim açısına dönüş yapmak için kullanılan değişkenin referansı.

FontNameGet

Yazı-tipi isim bilgisini alır.

```
string  FontNameGet();
```

Dönüş değeri

yazı tipi ismi

FontNameSet

Yazı tipi ismini ayarlar.

```
bool  FontNameSet(
   string  name      // isim
   );
```

Parametreler

name

[in] Yazı tipi ismi. Örneğin, "Arial".

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'

FontSet

Mevcut yazı tipini ayarlar.

```
bool  FontSet(
   const string  name,        // isim
   const int     size,        // boyut
   const uint    flags=0,     // bayraklar
   const uint    angle=0      // açı
   );
```

Parametreler

name

[in] Yazı tipi ismi. Örneğin, "Arial".

size

[in] Yazı-tipi boyutu. Boyut ayarlama hakkında daha fazla bilgi için TextSetFont() fonksiyonunun açıklamasına bakın.

flags=0

[in] Yazı-tipi oluşturma bayrakları. Bayraklar hakkında daha fazla bilgi için TextSetFont() fonksiyonunun açıklamasına bakın.

angle=0

[in] Ondalık değerler ile yazı-tipi eğim açısı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'

FontSizeGet

Yazı-tipi boyutunu alır.

```
int  FontSizeGet();
```

Dönüş değeri

yazı-tipi boyutu

FontSizeSet

Yazı-tipi boyutunu ayarlar.

```
bool  FontSizeSet(
   int  size      // boyut
   );
```

Parametreler

size

[in] Yazı-tipi boyutu. Boyut ayarlama hakkında daha fazla bilgi için TextSetFont() fonksiyonunun açıklamasına bakın.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'

Height

Bir grafiksel kaynağın yüksekliğini alır.

```
int  Height();
```

Dönüş değeri

grafiksel kaynağın yüksekliği

Line

Bir serbest-çizgi parçası çizer.

```
void  Line(
   int          x1,       // X koordinatı
   int          y1,       // Y koordinatı
   int          x2,       // X koordinatı
   int          y2,       // Y koordinatı
   const uint   clr       // renk
   );
```

Parametreler

x1

[in] Parçanın ilk noktasının X koordinatı.

y1

[in] Parçanın ilk noktasının Y koordinatı.

x2

[in] Parçanın ikinci noktasının X koordinatı.

y2

[in] Parçanın ikinci noktasının Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

LineAA

Antialiasing algoritması kullanarak bir serbest-çizgi parçası çizer.

```
void  LineAA(
   const int   x1,                  // X koordinatı
   const int   y1,                  // Y koordinatı
   const int   x2,                  // X koordinatı
   const int   y2,                  // Y koordinatı
   const uint  clr,                 // renk
   const uint  style=UINT_MAX       // çizgi stili
   );
```

Parametreler

x1

[in] Parçanın ilk noktasının X koordinatı.

y1

[in] Parçanın ilk noktasının Y koordinatı.

x2

[in] Parçanın ikinci noktasının X koordinatı.

y2

[in] Parçanın ikinci noktasının Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

style=UINT_MAX

[in] Çizgi stili, ENUM_LINE_STYLE sayımının değerlerinden biri veya kullanıcı tanımlı bir değer olabilir.

LineWu

Wu antialiasing algoritması kullanarak bir serbest-çizgi parçası çizer.

```
void  LineWu(
   const int   x1,                  // X koordinatı
   const int   y1,                  // Y koordinatı
   const int   x2,                  // X koordinatı
   const int   y2,                  // Y koordinatı
   const uint  clr,                 // renk
   const uint  style=UINT_MAX       // çizgi stili
   );
```

Parametreler

x1

[in] Parçanın ilk noktasının X koordinatı.

y1

[in] Parçanın ilk noktasının Y koordinatı.

x2

[in] Parçanın ikinci noktasının X koordinatı.

y2

[in] Parçanın ikinci noktasının Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

style=UINT_MAX

[in] Çizgi stili, ENUM_LINE_STYLE sayımının değerlerinden biri veya kullanıcı tanımlı bir değer olabilir.

LineHorizontal

Bir yatay çizgi parçası çizer.

```
void  LineHorizontal(
   int          x1,       // X koordinatı
   int          x2,       // X koordinatı
   int          y,        // Y koordinatı
   const uint   clr       // renk
   );
```

Parametreler

x1

[in] Parçanın ikinci noktasının X koordinatı.

x2

[in] Parçanın birinci noktasının X koordinatı.

y

[in] Parçanın ilk noktasının Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

LineVertical

Bir dikey çizgi parçası çizer.

```
void  LineVertical(
   int          x,        // X koordinatı
   int          y1,       // Y koordinatı
   int          y2,       // Y koordinatı
   const uint   clr       // renk
   );
```

Parametreler

x

[in] Parçanın X koordinatı.

y1

[in] Parçanın ilk noktasının Y koordinatı.

y2

[in] Parçanın ikinci noktasının Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

LineStyleSet

Çizgi stilini ayarlar.

```
void  LineStyleSet(
   const uint  style      // stil
   );
```

Parametreler

style

[in] Çizgi stili.

Not

Giriş parametreleri ENUM_LINE_STYLE sayımının değerlerinden herhangi birini alabilir. Ayrıca, kullanıcı tanımlı bir çizim stili oluşturmak da mümkündür.

LineThick

Antialiasing algoritması kullanarak belirtilen kalınlıkta bir serbest-çizgi parçası çizer.

```
void  LineThick(
   const int       x1,            // parçanın ilk noktasının X koordinatı
   const int       y1,            // parçanın ilk noktasının Y koordinatı
   const int       x2,            // parçanın ikinci noktasının X koordinatı
   const int       y2,            // parçanın ikinci noktasının Y koordinatı
   const uint      clr,           // renk
   const int       size,          // çizgi kalınlığı
   const uint      style,         // çizgi stili
   ENUM_LINE_END   end_style      // çizgi sonu stili
   )
```

Parametreler

x1

[in] Parçanın ilk noktasının X koordinatı.

y1

[in] Parçanın ilk noktasının Y koordinatı.

x2

[in] Parçanın ikinci noktasının X koordinatı.

y2

[in] Parçanın ikinci noktasının Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

size

[in] Çizgi genişliği.

style

[in] Çizgi stili ENUM_LINE_STYLE değerlerinden biri veya özel bir değer olabilir.

end_style

[in] Çizgi sonu stili ENUM_LINE_END sayımının değerlerini alabilir

ENUM_LINE_END

Tanıtıcı	Açıklama
LINE_END_ROUND	Çizgi sonu yuvarlanır.
LINE_END_BUTT	Çizgi sonu kesiktir.
LINE_END_SQUARE	Çizgi dolgulu dikdörtgen ile biter.

LineThickVertical

Antialiasing algoritması kullanarak belirtilen kalınlıkta bir dikey serbest-çizgi parçası çizer.

```
void  LineThickVertical(
   const int       x,              // parçanın X koordinatı
   const int       y1,             // parçanın ilk noktasının Y koordinatı
   const int       y2,             // parçanın ikinci noktasının Y koordinatı
   const uint      clr,            // renk
   const int       size,           // çizgi kalınlığı
   const uint      style,          // çizgi stili
   ENUM_LINE_END   end_style       // çizgi sonu stili
   )
```

Parametreler

x

[in] Parçanın X koordinatı.

y1

[in] Parçanın ilk noktasının Y koordinatı.

y2

[in] Parçanın ikinci noktasının Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

size

[in] Çizgi genişliği.

style

[in] Çizgi stili ENUM_LINE_STYLE değerlerinden biri veya özel bir değer olabilir.

end_style

[in] Çizgi sonu stili ENUM_LINE_END sayımının değerlerini alabilir.

LineThickHorizontal

Antialiasing algoritması kullanarak belirtilen kalınlıkta bir yatay serbest-çizgi parçası çizer.

```
void  LineThickHorizontal(
   const int       x1,            // parçanın ilk noktasının X koordinatı
   const int       x2,            // parçanın ikinci noktasının X koordinatı
   const int       y,             // parçanın Y koordinatı
   const uint      clr,           // renk
   const int       size,          // çizgi kalınlığı
   const uint      style,         // çizgi stili
   ENUM_LINE_END   end_style      // çizgi sonu stili
   )
```

Parametreler

x1

[in] Parçanın ilk noktasının X koordinatı.

x2

[in] Parçanın ikinci noktasının X koordinatı.

y

[in] Parçanın Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

size

[in] Çizgi genişliği.

style

[in] Çizgi stili ENUM_LINE_STYLE değerlerinden biri veya özel bir değer olabilir.

end_style

[in] Çizgi sonu stili ENUM_LINE_END sayımının değerlerini alabilir.

LoadFromFile

Bir BMP dosyasından görüntü okur.

```
bool  LoadFromFile(
   const string  filename      // dosya ismi
   );
```

Parametreler

filename

[in] Dosya ismi ("BMP" uzantısı dahil olacak şekilde).

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'

PixelGet

Belirtilen koordinatlardaki noktanın renk bilgisini alır.

```
uint  PixelGet(
   const int  x,      // X koordinatı
   const int  y       // Y koordinatı
   );
```

Parametreler

x

[in] Noktanın X koordinatı.

y

[in] Noktanın Y koordinatı.

Dönüş değeri

ARGB biçiminde noktanın rengi.

PixelSet

Belirtilen koordinatlardaki noktanın rengini ayarlar.

```
void  PixelSet(
   const int   x,       // X koordinatı
   const int   y,       // Y koordinatı
   const uint  clr      // renk
   );
```

Parametreler

x

[in] Noktanın X koordinatı.

y

[in] Noktanın Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

PixelSetAA

Antialiasing algoritması kullanarak bir nokta çizer.

```
void  PixelSetAA(
   const double  x,        // X koordinatı
   const double  y,        // Y koordinatı
   const uint    clr       // renk
   );
```

Parametreler

x

[in] Noktanın X koordinatı.

y

[in] Noktanın Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

Polygon

Bir çokgen çizer.

```
void  Polygon(
   int&         x[],     // X koordinatları dizisi
   int&         y[],     // Y koordinatları dizisi
   const uint   clr      // renk
   );
```

Parametreler

x[]

[in] Çokgen noktalarının X koordinatları dizisi.

y[]

[in] Çokgen noktalarının Y koordinatları dizisi.

clr

[in] ARGB biçiminde renk bilgisi.

PolygonAA

Antialiasing algoritması kullanarak bir çokgen çizer.

```
void  PolygonAA(
   int&         x[],                // X koordinatları dizisi
   int&         y[],                // Y koordinatları dizisi
   const uint   clr,                // renk
   const uint   style=UINT_MAX      // çizgi stili
   );
```

Parametreler

x[]

[in] Çokgen noktalarının X koordinatları dizisi.

y[]

[in] Çokgen noktalarının Y koordinatları dizisi.

clr

[in] ARGB biçiminde renk bilgisi.

style=UINT_MAX

[in] Çizgi stili, ENUM_LINE_STYLE sayımının değerlerinden biri veya kullanıcı tanımlı bir değer olabilir.

PolygonWu

Wu antialiasing algoritması kullanarak bir çokgen çizer.

```
void  PolygonWu(
   int&         x[],                // X koordinatları dizisi
   int&         y[],                // Y koordinatları dizisi
   const uint   clr,                // renk
   const uint   style=UINT_MAX      // çizgi stili
   );
```

Parametreler

x[]

[in] Çokgen noktalarının X koordinatları dizisi.

y[]

[in] Çokgen noktalarının Y koordinatları dizisi.

clr

[in] ARGB biçiminde renk bilgisi.

style=UINT_MAX

[in] Çizgi stili, ENUM_LINE_STYLE sayımının değerlerinden biri veya kullanıcı tanımlı bir değer olabilir.

PolygonThick

Antialiasing algoritması kullanarak belirtilen kalınlıkta bir çokgen çizer.

```
void  PolygonThick(
   const int&      x[],          // poligon noktalarının X koordinatları dizisi
   const int&      y[],          // poligon noktalarının Y koordinatları dizisi
   const uint      clr,          // renk
   const int       size,         // çizgi kalınlığı
   const uint      style,        // çizgi stili
   ENUM_LINE_END   end_style     // çizgi sonu stili
   )
```

Parametreler

x[]

[in] Poligon noktalarının X koordinatları dizisi.

y[]

[in] Poligon noktalarının Y koordinatları dizisi.

clr

[in] ARGB biçiminde renk bilgisi.

size

[in] Çizgi genişliği.

style

[in] Çizgi stili ENUM_LINE_STYLE değerlerinden biri veya özel bir değer olabilir.

end_style

[in] Çizgi sonu stili ENUM_LINE_END sayımının değerlerini alabilir.

PolygonSmooth

İki antialiasing algoritması kullanarak belirtilen kalınlıkta bir çokgen çizer.Önce, Parçalar ayrı olarak Beizer eğrilerine göre düzleştirilir. Sonra bu parçalardan oluşturulan çokgen, düzleştirme kalitesini artırmak için taramalı antialiasing algoritması uygulanır.

```
void  PolygonSmooth(
   int&             x[],                        // poligon noktalarının X koordinatl
   int&             y[],                        // poligon noktalarının Y koordinatl
   const uint       clr,                        // renk
   const int        size,                       // çizgi kalınlığı
   ENUM_LINE_STYLE  style=STYLE_SOLID,          // çizgi stili
   ENUM_LINE_END    end_style=LINE_END_ROUND,   // çizgi sonu stili
   double           tension=0.5,                // antialiasing parametresinin değe
   double           step=10                     // yakınsama çizgilerinin uzunluğu
   )
```

Parametreler

&x[]

[in] Poligon noktalarının X koordinatları dizisi.

&y[]

[in] Poligon noktalarının Y koordinatları dizisi.

clr

[in] ARGB biçiminde renk bilgisi.

size

[in] Çizgi genişliği.

style=STYLE_SOLID

[in] Çizgi stili ENUM_LINE_STYLE değerlerinden biri veya özel bir değer olabilir.

end_style=LINE_END_ROUND

[in] Çizgi sonu stili ENUM_LINE_END sayımının değerlerini alabilir.

tension=0.5

[in] Düzleştirme parametresinin değeri

step=10

[in] Yakınsama çizgilerinin uzunluğu.

Polyline

Devamlı (çoklu) çizgi çizer

```
void  Polyline(
   int&         x[],     // X koordinatları dizisi
   int&         y[],     // Y koordinatları dizisi
   const uint   clr      // renk
   );
```

Parametreler

x[]

[in] Devamlı-çizgi noktalarının X koordinatları dizisi.

y[]

[in] Devamlı-çizgi noktalarının Y koordinatları dizisi.

clr

[in] ARGB biçiminde renk bilgisi.

PolylineSmooth

Ardışık iki antialiasing algoritması kullanarak belirtilen kalınlıkta bir çoklu çizgi çizer. Önce, Parçalar ayrı olarak Beizer eğrilerine göre düzleştirilir. Sonra bu parçalardan oluşturulan çoklu çizgi, düzleştirme kalitesini artırmak için taramalı antialiasing algoritması uygulanır.

```
void  PolylineSmooth(
   const int&        x[],                        // çokgen noktalarının X koordinatları
   const int&        y[],                        // çokgen noktalarının Y koordinatları
   const uint        clr,                        // renk
   const int         size,                       // çizgi kalınlığı
   ENUM_LINE_STYLE   style=STYLE_SOLID,          // çizgi stili
   ENUM_LINE_END     end_style=LINE_END_ROUND,   // çizgi sonu stili
   double            tension=0.5,                // antialiasing parametresinin değeri
   double            step=10                     // yakınsama adımı
   )
```

Parametreler

&x[]

[in] Devamlı-çizgi noktalarının X koordinatları dizisi.

&y[]

[in] Devamlı-çizgi noktalarının Y koordinatları dizisi.

clr

[in] ARGB biçiminde renk bilgisi.

size

[in] Çizgi genişliği.

style=STYLE_SOLID

[in] Çizgi stili ENUM_LINE_STYLE değerlerinden biri veya özel bir değer olabilir.

end_style=LINE_END_ROUND

[in] Çizgi sonu stili ENUM_LINE_END sayımının değerlerini alabilir.

tension=0.5

[in] Düzleştirme parametresinin değeri

step=10

[in] Yakınsama adımı.

PolylineThick

Antialiasing algoritması kullanarak belirtilen kalınlıkta bir çoklu çizgi çizer.

```
void  PolylineThick(
   const int       &x[],          // çokgen noktalarının X koordinatları dizisi
   const int       &y[],          // çokgen noktalarının Y koordinatları dizisi
   const uint      clr,           // renk
   const int       size,          // çizgi kalınlığı
   const uint      style,         // çizgi stili
   ENUM_LINE_END   end_style      // çizgi sonu stili
   )
```

Parametreler

&x[]

[in] Devamlı-çizgi noktalarının X koordinatları dizisi.

&y[]

[in] Devamlı-çizgi noktalarının Y koordinatları dizisi.

clr

[in] ARGB biçiminde renk bilgisi.

size

[in] Çizgi genişliği.

style

[in] Çizgi stili ENUM_LINE_STYLE değerlerinden biri veya özel bir değer olabilir.

end_style

[in] Çizgi sonu stili ENUM_LINE_END sayımının değerlerini alabilir.

PolylineWu

Wu antialiasing algoritması kullanarak bir devamlı-çizgi çizer.

```
void  PolylineWu(
   int&         x[],                // X koordinatları dizisi
   int&         y[],                // Y koordinatları dizisi
   const uint   clr,                // renk
   const uint   style=UINT_MAX      // çizgi stili
   );
```

Parametreler

x[]

[in] Devamlı-çizgi noktalarının X koordinatları dizisi.

y[]

[in] Devamlı-çizgi noktalarının Y koordinatları dizisi.

clr

[in] ARGB biçiminde renk bilgisi.

style=UINT_MAX

[in] Çizgi stili, ENUM_LINE_STYLE sayımının değerlerinden biri veya kullanıcı tanımlı bir değer olabilir.

PolylineAA

Antialiasing algoritması kullanarak bir devamlı-çizgi çizer.

```
void  PolylineAA(
   int&         x[],                // X koordinatları dizisi
   int&         y[],                // Y koordinatları dizisi
   const uint   clr,                // renk
   const uint   style=UINT_MAX      // çizgi stili
   );
```

Parametreler

x[]

[in] Devamlı-çizgi noktalarının X koordinatları dizisi.

y[]

[in] Devamlı-çizgi noktalarının Y koordinatları dizisi.

clr

[in] ARGB biçiminde renk bilgisi.

style=UINT_MAX

[in] Çizgi stili, ENUM_LINE_STYLE sayımının değerlerinden biri veya kullanıcı tanımlı bir değer olabilir.

Rectangle

İki nokta kullanarak bir dikdörtgen çizer.

```
void  Rectangle(
   int          x1,      // X koordinatı
   int          y1,      // Y koordinatı
   int          x2,      // X koordinatı
   int          y2,      // Y koordinatı
   const uint   clr      // renk
   );
```

Parametreler

x1

[in] Birinci dikdörtgen şekillendirme noktasının X koordinatı.

y1

[in] Birinci dikdörtgen şekillendirme noktasının Y koordinatı.

x2

[in] İkinci dikdörtgen şekillendirme noktasının X koordinatı.

y2

[in] İkinci dikdörtgen şekillendirme noktasının Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

Resize

Bir grafiksel kaynağı yeniden boyutlandırır.

```
bool  Resize(
   const int  width,       // genişlik
   const int  height       // yükseklik
   );
```

Parametreler

width

[in] Grafiksel kaynağın yeni genişliği.

height

[in] Grafiksel kaynağın yeni yüksekliği.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'

Not

Yeniden boyutlandırma sırasında önceki görüntü kaydedilmez.

ResourceName

Bir grafiksel kaynağın isim bilgisini alır

```
string  ResourceName();
```

Dönüş değeri

grafiksel kaynağın ismi

TextHeight

Metnin yükseklik bilgisini alır.

```
int  TextHeight(
   const string  text      // metin
   );
```

Parametreler

text

[in] Ölçümlenecek metin.

Dönüş değeri

Piksel bazında metin yüksekliği

Not

Metni ölçümlemek için mevcut yazı tipi kullanılır.

TextOut

Metni görüntüler.

```
void  TextOut(
   int          x,               // X koordinatı
   int          y,               // Y koordinatı
   string       text,            // metin
   const uint   clr,             // renk
   uint         alignment=0      // hizalama
   );
```

Parametreler

x

[in] Metnin tutturma noktasının X koordinatı.

y

[in] Metnin tutturma noktasının Y koordinatı.

text

[in] Görüntülenecek metin.

clr

[in] ARGB biçiminde renk bilgisi.

alignment=0

[in] Metin sabitleme yöntemi. Sabitleme yöntemleri hakkında daha fazla bilgi için TextOut() fonksiyonunun açıklamasına bakın.

Not

Metni görüntülemek için mevcut yazı tipi kullanılır.

TextSize

Metnin boyut bilgisini alır.

```
void  TextSize(
   const string  text,        // metin
   int&          width,       // genişlik
   int&          height       // yükseklik
   );
```

Parametreler

text

[in] Ölçümlenecek metin.

width

[out] Metin genişliğine dönüş yapmak için kullanılan değişkenin referansı.

height

[out] Metin yüksekliğine dönüş yapmak için kullanılan değişkenin referansı.

Not

Metni ölçümlemek için mevcut yazı tipi kullanılır.

TextWidth

Metnin genişlik bilgisini alır.

```
int  TextWidth(
   const string  text      // metin
   );
```

Parametreler

text

[in] Ölçümlenecek metin.

Dönüş değeri

Piksel bazında metin yüksekliği

Not

Metni ölçümlemek için mevcut yazı tipi kullanılır.

TransparentLevelSet

Saydamlık seviyesini ayarlar.

```
void  TransparentLevelSet(
   const uchar  value      // değer
   );
```

Parametreler

value

[in] Saydamlık seviyesinin yeni değeri.

Not

Tamamen saydamlık için 0 değeri, bütünüyle opaklık için ise 255 değeri kullanılır.

Saydamlık seviyesinin ayarlanması önceki tüm çizimleri etkilerken daha sonradan yapılanları etkilemeyecektir.

Triangle

Bir üçgen çizer.

```
void  Triangle(
   int          x1,      // X koordinatı
   int          y1,      // Y koordinatı
   int          x2,      // X koordinatı
   int          y2,      // Y koordinatı
   int          x3,      // X koordinatı
   int          y3,      // Y koordinatı
   const uint   clr      // renk
   );
```

Parametreler

x1

[in] Üçgenin ilk köşesinin X koordinatı.

y1

[in] Üçgenin ilk köşesinin Y koordinatı.

x2

[in] Üçgenin ikinci köşesinin X koordinatı.

y2

[in] Üçgenin ikinci köşesinin Y koordinatı.

x3

[in] Üçgenin üçüncü köşesinin X koordinatı.

y3

[in] Üçgenin üçüncü köşesinin Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

TriangleAA

Antialiasing algoritması kullanarak bir dikdörtgen çizer.

```
void  TriangleAA(
   const int   x1,                  // X koordinatı
   const int   y1,                  // Y koordinatı
   const int   x2,                  // X koordinatı
   const int   y2,                  // Y koordinatı
   const int   x3,                  // X koordinatı
   const int   y3,                  // Y koordinatı
   const uint  clr,                 // renk
   const uint  style=UINT_MAX       // çizgi stili
   );
```

Parametreler

x1

[in] Üçgenin ilk köşesinin X koordinatı.

y1

[in] Üçgenin ilk köşesinin Y koordinatı.

x2

[in] Üçgenin ikinci köşesinin X koordinatı.

y2

[in] Üçgenin ikinci köşesinin Y koordinatı.

x3

[in] Üçgenin üçüncü köşesinin X koordinatı.

y3

[in] Üçgenin üçüncü köşesinin Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

style=UINT_MAX

[in] Çizgi stili, ENUM_LINE_STYLE sayımının değerlerinden biri veya kullanıcı tanımlı bir değer olabilir.

TriangleWu

Wu antialiasing algoritması kullanarak bir dikdörtgen çizer.

```
void  TriangleWu(
   const int   x1,                  // X koordinatı
   const int   y1,                  // Y koordinatı
   const int   x2,                  // X koordinatı
   const int   y2,                  // Y koordinatı
   const int   x3,                  // X koordinatı
   const int   y3,                  // Y koordinatı
   const uint  clr,                 // renk
   const uint  style=UINT_MAX       // çizgi stili
   );
```

Parametreler

x1

[in] Üçgenin ilk köşesinin X koordinatı.

y1

[in] Üçgenin ilk köşesinin Y koordinatı.

x2

[in] Üçgenin ikinci köşesinin X koordinatı.

y2

[in] Üçgenin ikinci köşesinin Y koordinatı.

x3

[in] Üçgenin üçüncü köşesinin X koordinatı.

y3

[in] Üçgenin üçüncü köşesinin Y koordinatı.

clr

[in] ARGB biçiminde renk bilgisi.

style=UINT_MAX

[in] Çizgi stili, ENUM_LINE_STYLE sayımının değerlerinden biri veya kullanıcı tanımlı bir değer olabilir.

Update

Değişiklikleri ekranda görüntüler.

```
void  Update(
   const bool  redraw=true      // bayrak
   );
```

Parametreler

redraw=true

Çizelge yenileme gerekliliğinin bayrağı.

Width

Bir grafiksel kaynağın genişlik bilgisini alır.

```
int  Width();
```

Dönüş değeri

grafiksel kaynağın genişliği

CChartCanvas

Çizelge ve çizelge bileşenleri oluşturmak için tasarlanan uygulama sınıfları için temel sınıf.

Açıklama

Bu sınıf, temel çizelge bileşenleriyle çalışmak için çeşitli yöntemler içerir: Koordinat eksenleri ve ölçek işaretleri, seri isimleri, ızgara, arkaplan, vb. Bileşenlerle ilgili görüntüleme seçenekleri buradan özelleştirilebilir: görünürlük, metin rengi, vb.

Bildirim

```
class CChartCanvas : public CCanvas
```

Başlık

```
#include <Canvas\Charts\ChartCanvas.mqh>
```

Kalıtım hiyerarşisi

CCanvas

CChartCanvas

Doğrudan kalıtım

CHistogramChart, CLineChart, CPieChart

Sınıf yöntemleri

Yöntem	Eylem
ColorBackground	Arka-plan rengini alır/ayarlar.
ColorBorder	Kenarlık rengini alır/ayarlar.
ColorText	Metin rengini alır/ayarlar.
ColorGrid	Izgara rengini alır/ayarlar.
MaxData	İzin verilen en fazla veri (serisi) sayısını alır/ayarlar.
MaxDescrLen	Açıklamalar için izin verilen en büyük uzunluğu alır/ayarlar.
ShowFlags	Çizelge elemanlarının görünürlük bayrağının değerini alır/ayarlar.
IsShowLegend	Çizelge veri serisi başlıklarının görünürlük bayrağı değerine dönüş yapar.
IsShowScaleLeft	Sol veri ölçeğinin görünürlük bayrağı değerine dönüş yapar.

Yöntem	Eylem
IsShowScaleRight	Sağ veri ölçeğinin görünürlük bayrağının değerine dönüş yapar.
IsShowScaleTop	Üst veri ölçeğinin görünürlük bayrağının değerine dönüş yapar.
IsShowScaleBottom	Alt veri ölçeğinin görünürlük bayrağının değerine dönüş yapar.
IsShowGrid	Çizelge ızgarasının görünürlük bayrağının değerine dönüş yapar.
IsShowDescriptors	Çizelge açıklamalarının görünürlük bayrağının değerine dönüş yapar.
IsShowPercent	Çizelgedeki yüzde değerlerinin görünürlük bayrağının değerine dönüş yapar.
VScaleMin	Dikey veri ölçeği için en küçük değeri alır/ayarlar.
VScaleMax	Dikey veri ölçeği için en büyük değeri alır/ayarlar.
NumGrid	Çizelge ızgarasının çizimi için dikey ölçek aralıklarının sayısını alır/ayarlar.
DataOffset	Veri konumunu alır/ayarlar.
DataTotal	Çizelgedeki veri serilerinin toplam sayısını alır.
DrawDescriptors	Açıklayıcıların çizimi için kullanılan sanal yöntem.
DrawData	Belirtilen indise sahip veri serilerini çizmek için sanal yöntem.
Create	Grafiksel kaynağı oluşturan sanal yöntem.
AllowedShowFlags	Çizelge bileşenleri için izin verilen görünürlük bayraklarını ayarlar.
ShowLegend	Veri açıklamalarının görünürlük bayrağını ayarlar.
ShowScaleLeft	Soldaki veri ölçeğinin görünürlük bayrağını ayarlar.
ShowScaleRight	Sağdaki veri ölçeğinin görünürlük bayrağını ayarlar.
ShowScaleTop	Üstteki veri ölçeğinin görünürlük bayrağını ayarlar.

Yöntem	Eylem
ShowScaleBottom	Alttaki veri ölçeğinin görünürlük bayrağını ayarlar.
ShowGrid	Izgaranın görünürlük bayrağını ayarlar.
ShowDescriptors	Açıklayıcıların görünürlük bayrağını ayarlar.
ShowValue	Değerlerin görünürlük bayrağını ayarlar.
ShowPercent	Yüzdelik değerlerin görünürlük bayrağını ayarlar.
LegendAlignment	Veri serilerinin açıklamaları için metin hizalamasını ayarlar.
Accumulative	Seriler için veri toplama bayrağını ayarlar.
VScaleParams	Dikey ölçek değerleri için parametreleri ayarlar.
DescriptorUpdate	Belirtilen pozisyondaki seri açıklayıcısının değerini günceller.
ColorUpdate	Belirtilen pozisyondaki serinin rengini günceller.
ValuesCheck	Çizelgeyi oluşturmak için içsel hesaplamalar gerçekleştirir.
Redraw	Çizelgeyi yeniden çizer.
DrawBackground	Arka-planı çizer.
DrawLegend	Veri serisi açıklamalarını çizer.
DrawLegendVertical	Veri serisi için dikey açıklama oluşturur.
DrawLegendHorizontal	Veri serisi için yatay açıklama oluşturur.
CalcScales	Ölçek koordinatlarını hesaplar.
DrawScales	Tüm veri ölçeklerini yeniden çizer.
DrawScaleLeft	Sol veri ölçeğini yeniden çizer.
DrawScaleRight	Sağ veri ölçeğini yeniden çizer.
DrawScaleTop	Üst veri ölçeğini yeniden çizer.
DrawScaleBottom	Alt veri ölçeğini yeniden çizer.
DrawGrid	Çizelgeyi yeniden çizer.
DrawChart	Çizelgeyi yeniden çizer.

CCanvas sınıfından türetilen yöntemler

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 (Get yöntemi)

Arka-plan rengine dönüş yapar.

```
uint  ColorBackground()
```

Dönüş Değeri

Background color.

ColorBackground (Set yöntemi)

Arka-plan rengini ayarlar.

```
void  ColorBackground(
  const uint  value,  // arka-plan rengi
  )
```

Parametreler

value

[in] Arkaplan rengi.

ColorBorder (Get yöntemi)

Kenarlık rengine dönüş yapar.

```
uint  ColorBorder()
```

Dönüş Değeri

Kenarlık rengi.

ColorBorder (Set yöntemi)

Kenarlık rengini ayarlar.

```
void  ColorBorder(
  const uint  value,  // kenarlık rengi
  )
```

Parametreler

value

[in] Kenarlık rengi.

ColorText (Get yöntemi)

Metin rengine dönüş yapar.

```
uint  ColorText()
```

Dönüş Değeri

Metin rengi

ColorText (Set yöntemi)

Metin rengini ayarlar

```
void  ColorText(
  const uint  value,  // metin rengi
  )
```

Parametreler

value

[in] Metin rengi.

ColorGrid (Get yöntemi)

Izgara rengine dönüş yapar.

```
uint  ColorGrid()
```

Dönüş Değeri

Izgara rengi.

ColorGrid (Set yöntemi)

Izgara rengini ayarlar.

```
void  ColorGrid(
  const uint  value,  // ızgara rengi
  )
```

Parametreler

value

[in] Izgara rengi.

MaxData (Get yöntemi)

İzin verilen maksimum veri miktarına dönüş yapar.

```
uint  MaxData()
```

Dönüş Değeri

Maksimum veri miktarı.

MaxData (Set yöntemi)

İzin verilen maksimum veri miktarını ayarlar.

```
void  MaxData(
  const uint  value,  // veri miktarı
  )
```

Parametreler

value

[in] Maksimum veri miktarı.

MaxDescrLen (Get yöntemi)

Açıklayıcılar için izin verilen en büyük uzunluğa dönüş yapar.

```
uint  MaxDescrLen()
```

Dönüş Değeri

Açıklayıcılar için izin verilen en büyük uzunluk.

MaxDescrLen (Set yöntemi)

Açıklayıcılar için izin verilen en büyük uzunluğu ayarlar.

```
void  MaxDescrLen(
  const uint  value,  // maksimum uzunluk
  )
```

Parametreler

value

[in] Açıklayıcılar için maksimum uzunluk.

ShowFlags (Get yöntemi)

Çizelge elemanlarının görünürlük bayrağının değerini alır.

```
bool  ShowFlags()
```

Dönüş Değeri

Çizelge elemanlarının görünürlük bayrağının değeri.

ShowFlags (Set yöntemi)

Çizelge elemanlarının görünürlük bayrağının değerini ayarlar.

```
void  ShowFlags(
  const uint  flags,  // bayrak
  )
```

Parametreler

flags

[in] Çizelge bileşenlerinin görünürlük değeri.

IsShowLegend

Çizelge veri serisi başlıklarının görünürlük bayrağı değerine dönüş yapar.

```
bool  IsShowLegend()
```

Dönüş Değeri

Başlıklar görünür ise 'true', aksi durumda 'false'.

IsShowScaleLeft

Sol veri ölçeğinin görünürlük bayrağı değerine dönüş yapar.

```
bool  IsShowScaleLeft()
```

Dönüş Değeri

Değer ölçeği görünür ise 'true', aksi durumda 'false'.

IsShowScaleRight

Sağ veri ölçeğinin görünürlük bayrağının değerine dönüş yapar.

```
bool  IsShowScaleRight()
```

Dönüş Değeri

Değer ölçeği görünür ise 'true', aksi durumda 'false'.

IsShowScaleTop

Üst veri ölçeğinin görünürlük bayrağının değerine dönüş yapar.

```
bool  IsShowScaleTop()
```

Dönüş Değeri

Değer ölçeği görünür ise 'true', aksi durumda 'false'.

IsShowScaleBottom

Alt veri ölçeğinin görünürlük bayrağının değerine dönüş yapar.

```
bool  IsShowScaleBottom()
```

Dönüş Değeri

Değer ölçeği görünür ise 'true', aksi durumda 'false'.

IsShowGrid

Çizelge ızgarasının görünürlük bayrağının değerine dönüş yapar.

```
bool  IsShowGrid()
```

Dönüş Değeri

Izgara görünür ise 'true', aksi durumda 'false'.

IsShowDescriptors

Çizelge açıklamalarının görünürlük bayrağının değerine dönüş yapar.

```
bool  IsShowDescriptors()
```

Dönüş Değeri

Açıklayıcılar görünür ise 'true', aksi durumda 'false'.

IsShowPercent

Çizelgedeki yüzde değerlerinin görünürlük bayrağının değerine dönüş yapar.

```
bool  IsShowPercent()
```

Dönüş Değeri

Yüzdeler görünür ise 'true', aksi durumda 'false'.

VScaleMin (Get yöntemi)

Dikey veri ölçeği için en küçük değere dönüş yapar.

```
double  VScaleMin()
```

Dönüş Değeri

Minimum dikey ölçek değeri.

VScaleMin (Set yöntemi)

Dikey veri ölçeği için en küçük değeri ayarlar.

```
void  VScaleMin(
  const double  value,    // dikey ölçek değeri
  )
```

Parametreler

value

[in] Minimum değer.

VScaleMax

Dikey veri ölçeği için en büyük değere dönüş yapar.

```
double  VScaleMax()
```

Dönüş Değeri

Maksimum dikey ölçek değeri.

VScaleMax

Dikey veri ölçeği için en büyük değeri ayarlar.

```
void  VScaleMax(
  const double  value,  //   dikey ölçek değeri
  )
```

Parametreler

value

[in] Maksimum değer.

NumGrid

Çizelge ızgarasının çizimi için dikey ölçek aralıklarının sayısını alır.

```
uint  NumGrid()
```

Dönüş Değeri

Aralık sayısı.

NumGrid

Çizelge ızgarasının çizimi için dikey ölçek aralıklarının sayısını ayarlar.

```
void  NumGrid(
  const uint  value,  // aralıkların sayısı
  )
```

Parametreler

value

[in] Bölüm sayısı.

DataOffset

Veri konumunun değerine dönüş yapar.

```
int  DataOffset()
```

Dönüş Değeri

Veri konumu.

DataOffset

Verinin konumunu ayarlar.

```
void  DataOffset(
  const int  value,  // konum
  )
```

Parametreler

value

[in] Veri konumu.

DataTotal

Çizelgedeki veri serilerinin toplam sayısını alır.

```
uint  DataTotal()
```

Dönüş Değeri

Serilerin sayısı.

DrawDescriptors

Açıklayıcıların çizimi için kullanılan sanal yöntem.

```
virtual void  DrawDescriptors()
```

DrawData

Belirtilen indise sahip veri serilerini çizmek için sanal yöntem.

```
virtual void  DrawData(
   const uint  idx=0,  // indis
   )
```

Parametreler

idx=0

[in] Serinin indisi.

Create

Grafiksel kaynağı oluşturan sanal yöntem.

```
virtual bool  Create(
  const string       name,      // kaynak ismi
  const int          width,     // genişlik
  const int          height,    // yükseklik
  ENUM_COLOR_FORMAT  clrfmt,    // renk biçimi
  )
```

Parametreler

name

[in] Grafiksel kaynak ismi için temel kelime. Kaynak ismi, oluşturma işlemi sırasında kök kelimeye pseudo-rassal bir dizgi eklenerek meydana getirilir.

width

[in] Piksel bazında genişlik (X-ekseni üzerindeki boyut).

height

[in] Piksel bazında yükseklik (Y-ekseni üzerindeki boyut).

clrfmt

[in] Renk işleme yöntemi. Renk işleme yöntemleri hakkında daha fazla bilgi için ResourceCreate() fonksiyonunun açıklamasına bakın.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

AllowedShowFlags

Çizelge bileşenleri için izin verilen görünürlük bayraklarını ayarlar.

```
void  AllowedShowFlags(
  const uint  flags,  // bayraklar
  )
```

Parametreler

flags

[in] İzin verilen bayraklar.

ShowLegend

Veri serisi başlıklarının görünürlük bayrağını (FLAG_SHOW_LEGEND) ayarlar.

```
void  ShowLegend(
  const bool  flag,  // bayrak değeri
  )
```

Parametreler

flag

[in] Bayrak değeri:

- true — başlıklar görünür.
- false — başlıklar görünmez.

ShowScaleLeft

Sol veri ölçeğinin görünürlük bayrağını (FLAG_SHOW_SCALE_LEFT) ayarlar.

```
void  ShowScaleLeft(
  const bool  flag,  // bayrak değeri
  )
```

Parametreler

flag

[in] Bayrak değeri:

- true — sol ölçek görünür.
- false — sol ölçek görünmez.

ShowScaleRight

Sağ veri ölçeğinin görünürlük bayrağını (FLAG_SHOW_SCALE_RIGHT) ayarlar.

```
void  ShowScaleRight(
  const bool  flag,  // bayrak değeri
  )
```

Parametreler

flag

[in] Bayrak değeri:

- true — sağ ölçek görünür.
- false — sağ ölçek görünmez.

ShowScaleTop

Üst veri ölçeğinin görünürlük bayrağını (FLAG_SHOW_SCALE_TOP) ayarlar.

```
void  ShowScaleTop(
  const bool  flag,  // bayrak değeri
  )
```

Parametreler

flag

[in] Bayrak değeri:

- true — üst ölçek görünür.
- false — üst ölçek görünmez.

ShowScaleBottom

Alt veri ölçeğinin görünürlük bayrağını (FLAG_SHOW_SCALE_BOTTOM) ayarlar.

```
void  ShowScaleBottom(
  const bool  flag,     // bayrak değeri
  )
```

Parametreler

flag

[in] Bayrak değeri:

- true — alt ölçek görünür.
- false — alt ölçek görünmez.

ShowGrid

Izgara için görünürlük bayrağının (FLAG_SHOW_GRID) değerini alır.

```
void  ShowGrid(
  const bool  flag,  // bayrak değeri
  )
```

Parametreler

flag

[in] Bayrak değeri:

- true — ızgara görünür.
- false — ızgara görünmez.

ShowDescriptors

Açıklayıcıların görünürlük bayrağını (FLAG_SHOW_DESCRIPTORS) ayarlar.

```
void  ShowDescriptors(
  const bool  flag,    // bayrak değeri
  )
```

Parametreler

flag

[in] Bayrak değeri:

- true — açıklayıcı görünür.
- false — açıklayıcı görünmez.

ShowValue

Değerlerin görünürlük bayrağını (FLAG_SHOW_VALUE) ayarlar.

```
void  ShowValue(
  const bool  flag,  // bayrak değeri
  )
```

Parametreler

flag

[in] Bayrak değeri:

- true — değerler görünür.
- false — değerler görünmez.

ShowPercent

Yüzdelik değerlerin görünürlük bayrağını (FLAG_SHOW_PERCENT) ayarlar.

```
void  ShowPercent(
  const bool  flag,  // bayrak değeri
  )
```

Parametreler

flag

[in] Bayrak değeri:

- true — yüzdeler görünür.
- false — yüzdeler görünmez.

LegendAlignment

Veri serilerinin açıklamaları için metin hizalamasını ayarlar.

```
void  LegendAlignment(
  const ENUM_ALIGNMENT  value,  // bayrak
  )
```

Parametreler

value

[in] ENUM_ALIGNMENT sayımının değerlerinden birini alır:

- ALIGNMENT_LEFT — sola hizalama.
- ALIGNMENT_TOP — yukarı hizalama.
- ALIGNMENT_RIGHT — sağa hizalama.
- ALIGNMENT_BOTTOM — aşağı hizalama.

Accumulative

Seriler için veri toplama bayrağını ayarlar.

```
void  Accumulative(
  const bool  flag=true,  // bayrak değeri
  )
```

Parametreler

flag=true

[in] Bayrak değeri:

- true — serinin mevcut değeri tüm geçmiş verilerin toplamıyla değiştirilir.
- false — serilerin çizimi için standart mod.

VScaleParams

Dikey ölçek değerleri için parametreleri ayarlar.

```
void  VScaleParams(
  const double  max,   // maksimum
  const double  min,   // minimum
  const uint    grid,  // bölümlerin sayısı
  )
```

Parametreler

max

[in] Minimum değer.

min

[in] Maksimum değer.

grid

[in] Ölçek bölümlerinin sayısı.

DescriptorUpdate

Belirtilen pozisyondaki seri açıklayıcısının değerini günceller.

```
bool  DescriptorUpdate(
  const uint    pos,    // indis
  const string  descr,  // değer
  )
```

Parametreler

pos

[in] Serinin indisi (serilerin numaralandırılmasına 0 ile başlanır).

descr

[in] Açıklayıcı değeri.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

ColorUpdate

Belirtilen pozisyondaki serinin rengini günceller.

```
bool  ColorUpdate(
  const uint  pos,  // indis
  const uint  clr,  // renk
  )
```

Parametreler

pos

[in] Serinin indisi (serilerin numaralandırılmasına 0 ile başlanır).

clr

[in] Renk değeri.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

ValuesCheck

Çizelgenin oluşturulması için içsel hesaplamaları yapan yardımcı sanal yöntem.

```
virtual void  ValuesCheck()
```

Redraw

Çizelgenin yeniden çizimi için kullanılan sanal yöntem.

```
virtual void  Redraw()
```

DrawBackground

Arka-plan çizimi için kullanılan sanal yöntem.

```
virtual void  DrawBackground()
```

DrawLegend

Veri serisi açıklamalarının çizimi için sanal yöntem.

```
virtual void  DrawLegend()
```

DrawLegendVertical

Veri serisi için dikey açıklama oluşturur.

```
int  DrawLegendVertical(
  const int  w,  // genişlik
  const int  h,  // yükseklik
  )
```

Parametreler

w

[in] Veri serisinin açıklaması için maksimum metin genişliği.

h

[in] Veri serisinin açıklaması için maksimum metin yüksekliği.

Dönüş Değeri

Veri serisi açıklamasının piksel cinsinden genişliği.

DrawLegendHorizontal

Veri serisi için yatay açıklama oluşturur.

```
int  DrawLegendHorizontal(
  const int  w,  //
  const int  h,  //
  )
```

Parametreler

w

[in] Veri serisinin açıklaması için maksimum metin genişliği.

h

[in] Veri serisinin açıklaması için maksimum metin yüksekliği.

Dönüş Değeri

Veri serisi açıklamasının piksel cinsinden yüksekliği.

CalcScales

Ölçek değerleri için etiketlerin konumunu hesaplayan sanal yöntem.

```
virtual void  CalcScales()
```

DrawScales

Tüm değer ölçeklerini çizmek için kullanılan sanal yöntem.

```
virtual void  DrawScales()
```

DrawScaleLeft

Sol ölçek çizimi için kullanılan sanal yöntem.

```
virtual int  DrawScaleLeft(
  const bool  draw,  // bayrak
  )
```

Parametreler

draw

[in] Ölçeğin yeniden çizilip çizilmeyeceğini belirten bayrak değeri.

Dönüş Değeri

Değer ölçeğinin genişliği.

DrawScaleRight

Sağ ölçek çizimi için kullanılan sanal yöntem.

```
virtual int  DrawScaleRight(
  const bool  draw,  // bayrak
  )
```

Parametreler

draw

[in] Ölçeğin yeniden çizilip çizilmeyeceğini belirten bayrak değeri.

Dönüş Değeri

Değer ölçeğinin genişliği.

DrawScaleTop

Üst ölçek çizimi için kullanılan sanal yöntem.

```
virtual int  DrawScaleTop(
  const bool  draw,  // bayrak
  )
```

Parametreler

draw

[in] Ölçeğin yeniden çizilip çizilmeyeceğini belirten bayrak değeri.

Dönüş Değeri

Değer ölçeğinin yüksekliği.

DrawScaleBottom

Alt ölçek çizimi için kullanılan sanal yöntem.

```
virtual int  DrawScaleBottom(
  const bool  draw,  // bayrak
  )
```

Parametreler

draw

[in] Ölçeğin yeniden çizilip çizilmeyeceğini belirten bayrak değeri.

Dönüş Değeri

Değer ölçeğinin yüksekliği.

DrawGrid

Izgaranın çizimi için kullanılan sanal yöntem.

```
virtual void  DrawGrid()
```

DrawChart

Çizelgenin yeniden çizimi için kullanılan sanal yöntem.

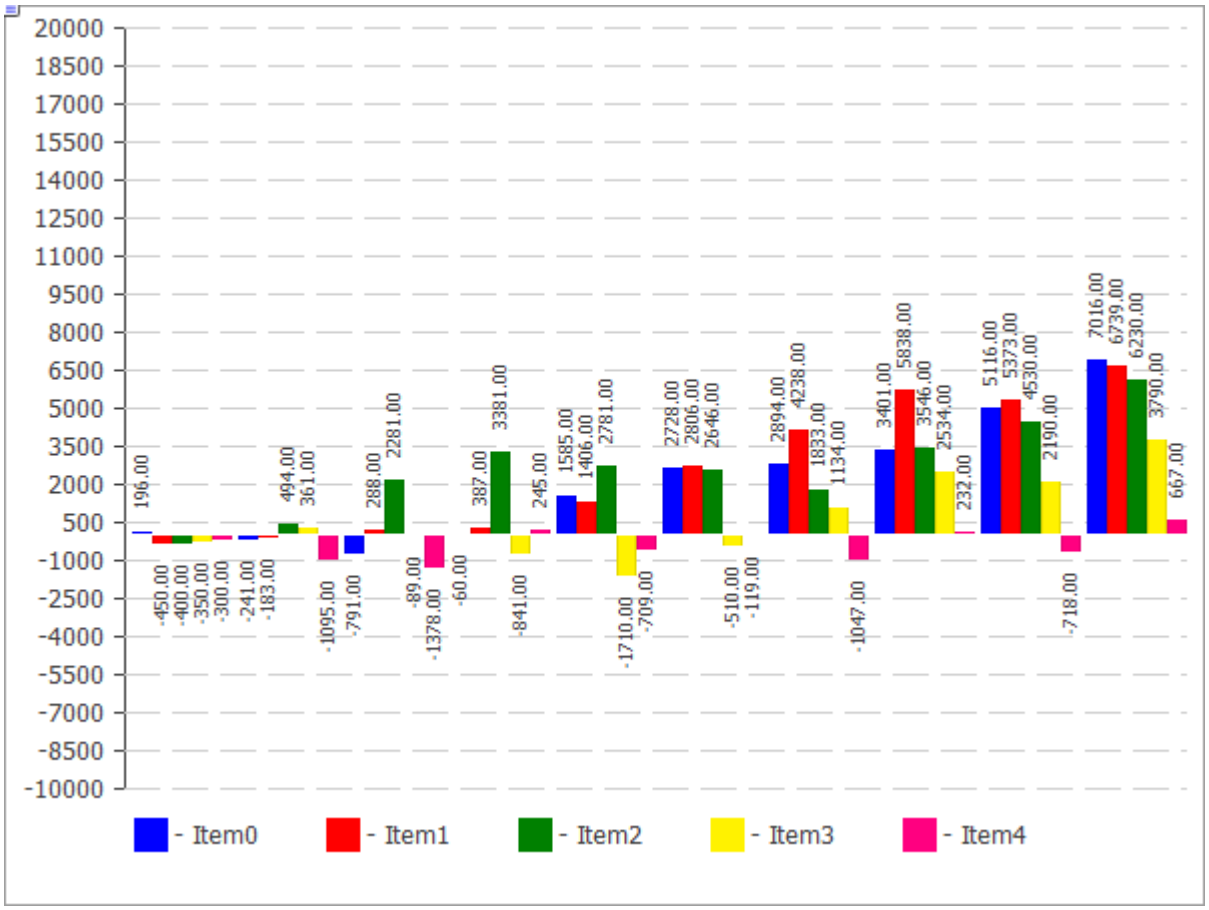
```
virtual void  DrawChart()
```

CHistogramChart

Histogram çizimi için tasarlanmıştır.

Açıklama

Histogram çizimi için kullanılan tüm yöntemler bu sınıfta yer alır. Bunlar verileri ve çubuk kalınlığını ayarlamak için kullanılabilir. Ayrıca verilerin daha iyi görüntülenebilmesi için desenli desenli çubuk dolguları da yöntemlere eklenmiştir.



Yukarıdaki şekli oluşturmak için kullanılan kod aşağıda paylaşılmıştır.

Bildirim

```
class CHistogramChart : public CChartCanvas
```

Başlık

```
#include <Canvas\Charts\HistogramChart.mqh>
```

Kalıtım hiyerarşisi

CCanvas

CChartCanvas

CHistogramChart

Sınıf yöntemleri

Yöntem	Eylem
Gradient	Histogram çubukları için desenli dolgunun uygulanma durumunu gösteren bayrağı ayarlar.
BarGap	Histogramın orjine göre girintisini ayarlar.
BarMinSize	Histogram çubuklarının en küçük genişliğini ayarlar.
BarBorder	Histogram çubuklarına kenarlık çizilip çizilmeyeceğini gösteren bayrağı ayarlar.
Create	Grafiksel kaynağı oluşturan sanal yöntem.
SeriesAdd	Yeni veri serisi ekler.
SeriesInsert	Çizelgeye veri serileri ekler.
SeriesUpdate	Çizelgedeki veri serilerini günceller.
SeriesDelete	Çizelgedeki veri serilerini siler.
ValueUpdate	Belirtilen serideki bileşenin değerini günceller.
DrawData	Belli bir seri için histogram oluşturan sanal yöntem.
DrawBar	Histogram çubuklarını dolgulu dikdörtgen şeklinde çizer.
GradientBrush	Desenli dolgu için bir fırça seçer.

CCanvas sınıfından türetilen yöntemler

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

CChartCanvas sınıfından türetilen yöntemler

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

Örnek

```
//+------------------------------------------------------------------+
//|                                          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 "Histogram kullanımına örnek"
//---
#include <Canvas\Charts\HistogramChart.mqh>
//+------------------------------------------------------------------+
//| inputs                                                           |
//+------------------------------------------------------------------+
input bool Accumulative=true;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   int k=100;
   double arr[10];
//--- çizelge oluştur
   CHistogramChart chart;
   if(!chart.CreateBitmapLabel("SampleHistogramChart",10,10,600,450))
     {
      Print("Hata, histogram oluşturulamadı: ",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));
     }
//--- değerlerle oyna
   while(!IsStopped())
     {
      int i=rand()%5;
      int j=rand()%10;
      k=rand()%3000-1000;
      chart.ValueUpdate(i,j,k);
      Sleep(200);
     }
```

```
//--- bitir
   chart.Destroy();
   return(0);
  }
```

Gradient

Histogram çubukları için desenli dolgunun uygulanma durumunu gösteren bayrağı ayarlar.

```
void  Gradient(
  const bool  flag=true,  // bayrak değeri
  )
```

Parametreler

flag=true

Bayrak değeri: desenli dolgu etkinse 'true', değilse 'false'.

BarGap

Histogramın orjine göre girintisini ayarlar.

```
void  BarGap(
  const uint  value,  // konum
  )
```

Parametreler

value

[in] Histogram konumu.

BarMinSize

Histogram çubuklarının en küçük genişliğini ayarlar.

```
void  BarMinSize(
  const uint  value,  // minimum genişlik
  )
```

Parametreler

value

[in] Minimum genişlik.

BarBorder

Histogram çubuklarına kenarlık çizilip çizilmeyeceğini gösteren bayrağı ayarlar.

```
void  BarBorder(
  const uint  value,  // bayrak
  )
```

Parametreler

value

[in] Bayrak değeri:

- true — kenarlıklar çizilir
- false — kenarlıklar çizilmez

Create

Grafiksel kaynağı oluşturan sanal yöntem.

```
virtual bool  Create(
  const string       name,    // isim
  const int          width,   // genişlik
  const int          height,  // yükseklik
  ENUM_COLOR_FORMAT  clrfmt,  // renk biçimi
  )
```

Parametreler

name

[in] Grafiksel kaynak ismi için temel kelime. Kaynak ismi, oluşturma işlemi sırasında kök kelimeye pseudo-rassal bir dizgi eklenerek meydana getirilir.

width

[in] Piksel bazında genişlik (X-ekseni üzerindeki boyut).

height

[in] Piksel bazında yükseklik (Y-ekseni üzerindeki boyut).

clrfmt

[in] Renk işleme yöntemi. Renk işleme yöntemleri hakkında daha fazla bilgi için ResourceCreate() fonksiyonunun açıklamasına bakın.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

SeriesAdd

Yeni veri serisi ekler.

```
bool  SeriesAdd(
  const double&  value[],  // değerler
  const string   descr,    // etiket
  const uint     clr,      // renk
  )
```

Parametreler

value[]

[in] Veri serisi.

descr

[in] Seri etiketi.

clr

[in] Serinin görüntü rengi.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

SeriesInsert

Çizelgeye veri serileri ekler.

```
bool  SeriesInsert(
  const uint    pos,        // indis
  const double&  value[],   // değerler
  const string  descr,      // etiket
  const uint    clr,        // renk
  )
```

Parametreler

pos

[in] Giriş indisi.

value[]

[in] Veri serisi.

descr

[in] Seri etiketi.

clr

[in] Serinin görüntü rengi.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

SeriesUpdate

Çizelgedeki veri serilerini günceller.

```
bool  SeriesUpdate(
  const uint    pos,        // indis
  const double  &value[],   // değerler
  const string  descr,      // etiket
  const uint    clr,        // renk
  )
```

Parametreler

pos

[in] Serinin indisi (serilerin numaralandırılmasına 0 ile başlanır).

&value[]

[in] Veri serisinin yeni rengi.

descr

[in] Seri etiketi.

clr

[in] Serinin görüntü rengi.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

SeriesDelete

Çizelgedeki veri serilerini siler.

```
bool  SeriesDelete(
  const uint  pos,  // indis
  )
```

Parametreler

pos

[in] Serinin indisi (serilerin numaralandırılmasına 0 ile başlanır).

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

ValueUpdate

Belirtilen seride belirtilen elemanın değerini günceller.

```
bool  ValueUpdate(
  const uint  series,  // indis of the series
  const uint  pos,     // indis of the element
  double      value,   // değer
  )
```

Parametreler

series

[in] Serinin indisi (serilerin numaralandırılmasına 0 ile başlanır).

pos

[in] Verinin seri içindeki indisi.

value

[in] Yeni değer.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

DrawData

Belli bir seri için histogram oluşturan sanal yöntem.

```
virtual void  DrawData(
  const uint  index,  // indis
  )
```

Parametreler

index

[in] Serinin indisi (serilerin numaralandırılmasına 0 ile başlanır).

DrawBar

Histogram çubuklarını dolgulu dikdörtgen şeklinde çizer.

```
void  DrawBar(
  const int   x,    // X koordinatı
  const int   y,    // Y koordinatı
  const int   w,    // genişlik
  const int   h,    // yükseklik
  const uint  clr,  // renk
  )
```

Parametreler

x

[in] Dikdörtgenin sol üst köşesinin X koordinatı.

y

[in] Dikdörtgenin sol üst köşesinin Y koordinatı.

w

[in] Dikdörtgen genişliği.

h

[in] Dikdörtgen yüksekliği.

clr

[in] Dikdörtgenin rengi.

GradientBrush

Desenli dolgu için bir fırça seçer.

```
void  GradientBrush(
  const int   size,      // boyut
  const uint  fill_clr,  // dolgu rengi
  )
```

Parametreler

size

[in] Fırça kalınlığı

fill_clr

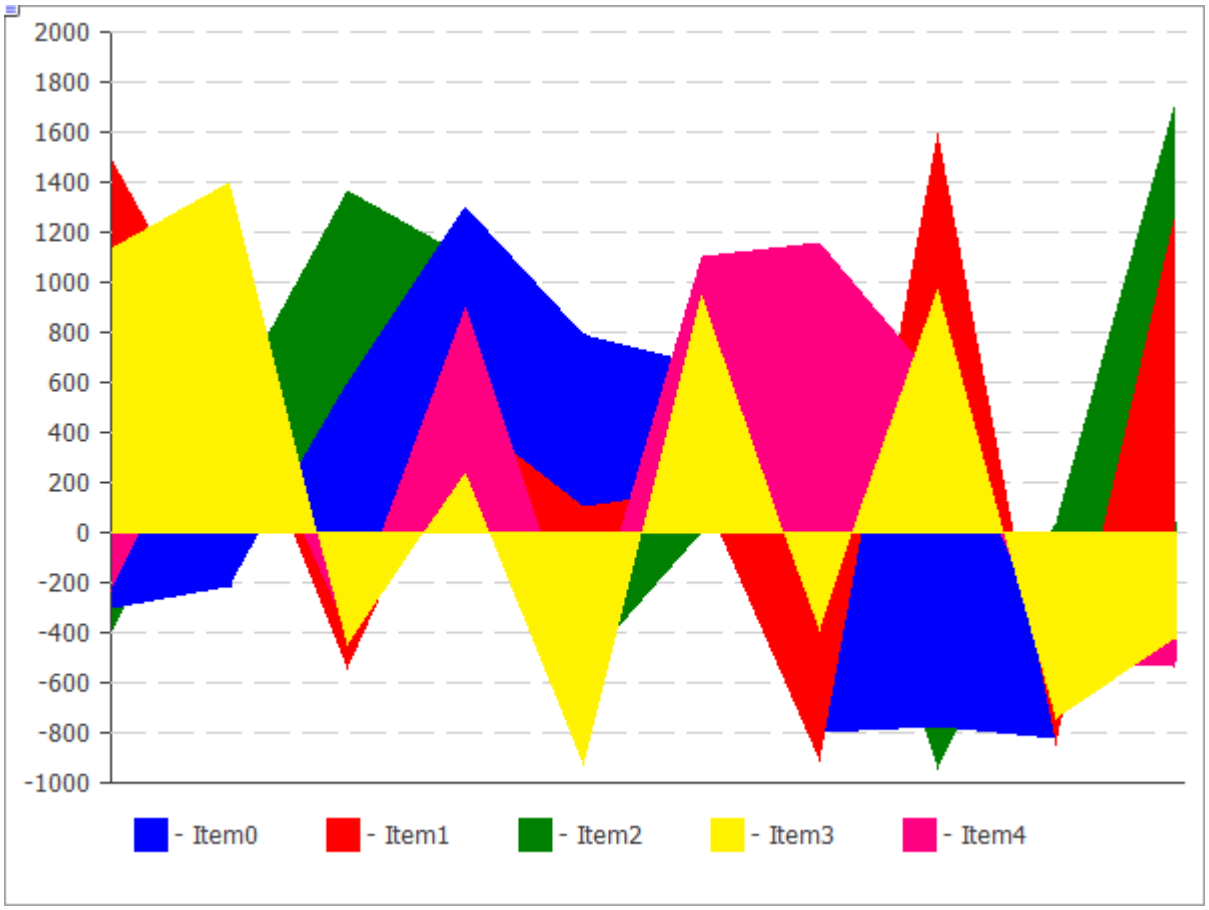
[in] Dolgu rengi

CLineChart

Eğri çizimi için tasarlanmıştır.

Açıklama

Sınıf içinde yer alan yöntemler çizelge üzerinde eğrilerle çalışmak için geliştirilmiştir. Çizilen eğriyle sınırlanan alana dolgu yapabilme özelliğiyle ön plana çıkar.



Yukarıdaki şekli oluşturmak için kullanılan kod aşağıda paylaşılmıştır.

Bildirim

```
class CLineChart : public CChartCanvas
```

Başlık

```
#include <Canvas\Charts\LineChart.mqh>
```

Kalıtım hiyerarşisi

CCanvas

CChartCanvas

CLineChart

Sınıf yöntemleri

Yöntem	Eylem
Filled	Eğriyle sınırlanan alan için dolgu oluşturma bayrağını ayarlar.
Create	Bir grafiksel kaynak oluşturur.
SeriesAdd	Yeni veri serisi ekler.
SeriesInsert	Çizelgeye veri serileri ekler.
SeriesUpdate	Çizelgedeki veri serilerini günceller.
SeriesDelete	Çizelgedeki veri serilerini siler.
ValueUpdate	Belirtilen seride belirtilen elemanın değerini günceller.
DrawChart	Tüm bileşenleriyle birlikte bir eğri çizmek için tasarlanmış sanal yöntem.
DrawData	Belli bir veri serisinin eğrisini çizmek için tasarlanmış sanal yöntem.
CalcArea	Veri setine göre çizilen eğrinin altında kalan alanı hesaplar.

CCanvas sınıfından türetilen yöntemler

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

CChartCanvas sınıfından türetilen yöntemler

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

Örnek

```
//+------------------------------------------------------------------+
//|                                                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 "Çizgi grafik kullanım örneği"
//---
#include <Canvas\Charts\LineChart.mqh>
//+------------------------------------------------------------------+
//| inputs                                                           |
//+------------------------------------------------------------------+
input bool Accumulative=false;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
   int k=100;
   double arr[10];
//--- çizelge oluştur
   CLineChart chart;
//--- çizelge oluştur
   if(!chart.CreateBitmapLabel("SampleHistogrammChart",10,10,600,450))
     {
      Print("Hata, çizgi grafiği oluşturulamadı: ",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));
     }
//--- değerlerle oyna
   while(!IsStopped())
     {
      int i=rand()%5;
      int j=rand()%10;
      k=rand()%3000-1000;
      chart.ValueUpdate(i,j,k);
      Sleep(200);
     }
```

```
//--- bitir
   chart.Destroy();
   return(0);
  }
```

Filled

Eğriyle sınırlanan alan için dolgu oluşturma gerekliliğini gösteren bayrağın değerini ayarlar.

```
void  Filled(
  const bool  flag=true,  // bayrak
  )
```

Parametreler

flag=true

[in] Bayrak değeri:

- true — eğri altında kalan alanı doldur
- false — eğri altında kalan alanı doldurma

Create

Grafiksel kaynağı oluşturan sanal yöntem.

```
virtual bool  Create(
  const string       name,    // isim
  const int          width,   // genişlik
  const int          height,  // yükseklik
  ENUM_COLOR_FORMAT  clrfmt,  // renk biçimi
  )
```

Parametreler

name

[in] Grafiksel kaynak ismi için temel kelime. Kaynak ismi, oluşturma işlemi sırasında kök kelimeye pseudo-rassal bir dizgi eklenerek meydana getirilir.

width

[in] Piksel bazında genişlik (X-ekseni üzerindeki boyut).

height

[in] Piksel bazında yükseklik (Y-ekseni üzerindeki boyut).

clrfmt

[in] Renk işleme yöntemi. Renk işleme yöntemleri hakkında daha fazla bilgi için ResourceCreate() fonksiyonunun açıklamasına bakın.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

SeriesAdd

Yeni veri serisi ekler.

```
bool  SeriesAdd(
  const double&  value[],  // değerler
  const string   descr,    // etiket
  const uint     clr,      // renk
  )
```

Parametreler

value[]

[in] Veri serisi.

descr

[in] Seri etiketi.

clr

[in] Serinin görüntü rengi.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

SeriesInsert

Çizelgeye veri serileri ekler.

```
bool  SeriesInsert(
  const uint     pos,        // indis
  const double&  value[],    // değerler
  const string   descr,      // etiket
  const uint     clr,        // renk
  )
```

Parametreler

pos

[in] Giriş indisi.

value[]

[in] Veri serisi.

descr

[in] Seri etiketi.

clr

[in] Serinin görüntü rengi.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

SeriesUpdate

Çizelgedeki veri serilerini günceller.

```
bool  SeriesUpdate(
  const uint     pos,        // indis
  const double&  value[],    // değerler
  const string   descr,      // etiket
  const uint     clr,        // renk
  )
```

Parametreler

pos

[in] Serinin indisi (serilerin numaralandırılmasına 0 ile başlanır).

value[]

[in] Veri serisinin yeni rengi.

descr

[in] Seri etiketi.

clr

[in] Serinin görüntü rengi.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

SeriesDelete

Çizelgedeki veri serilerini siler.

```
bool  SeriesDelete(
  const uint  pos,  // indis
  )
```

Parametreler

pos

[in] Serinin indisi (serilerin numaralandırılmasına 0 ile başlanır).

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

ValueUpdate

Belirtilen seride belirtilen elemanın değerini günceller.

```
bool  ValueUpdate(
  const uint  series,  // indis of the series
  const uint  pos,     // indis of the element
  double      value,   // değer
  )
```

Parametreler

series

[in] Serinin indisi (serilerin numaralandırılmasına 0 ile başlanır).

pos

[in] Verinin seri içindeki indisi.

value

[in] Yeni değer.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

DrawChart

Tüm bileşenleriyle birlikte bir eğri çizmek için tasarlanmış sanal yöntem.

```
virtual void  DrawChart()
```

DrawData

Belli bir veri serisinin eğrisini çizmek için tasarlanmış sanal yöntem.

```
virtual void  DrawData(
  const uint  index,  // indis
  )
```

Parametreler

index

[in] Serinin indisi (serilerin numaralandırılmasına 0 ile başlanır).

CalcArea

Veri setine göre çizilen eğrinin altında kalan alanı hesaplar.

```
double  CalcArea(
  const uint  index,  // indis
  )
```

Parametreler

index

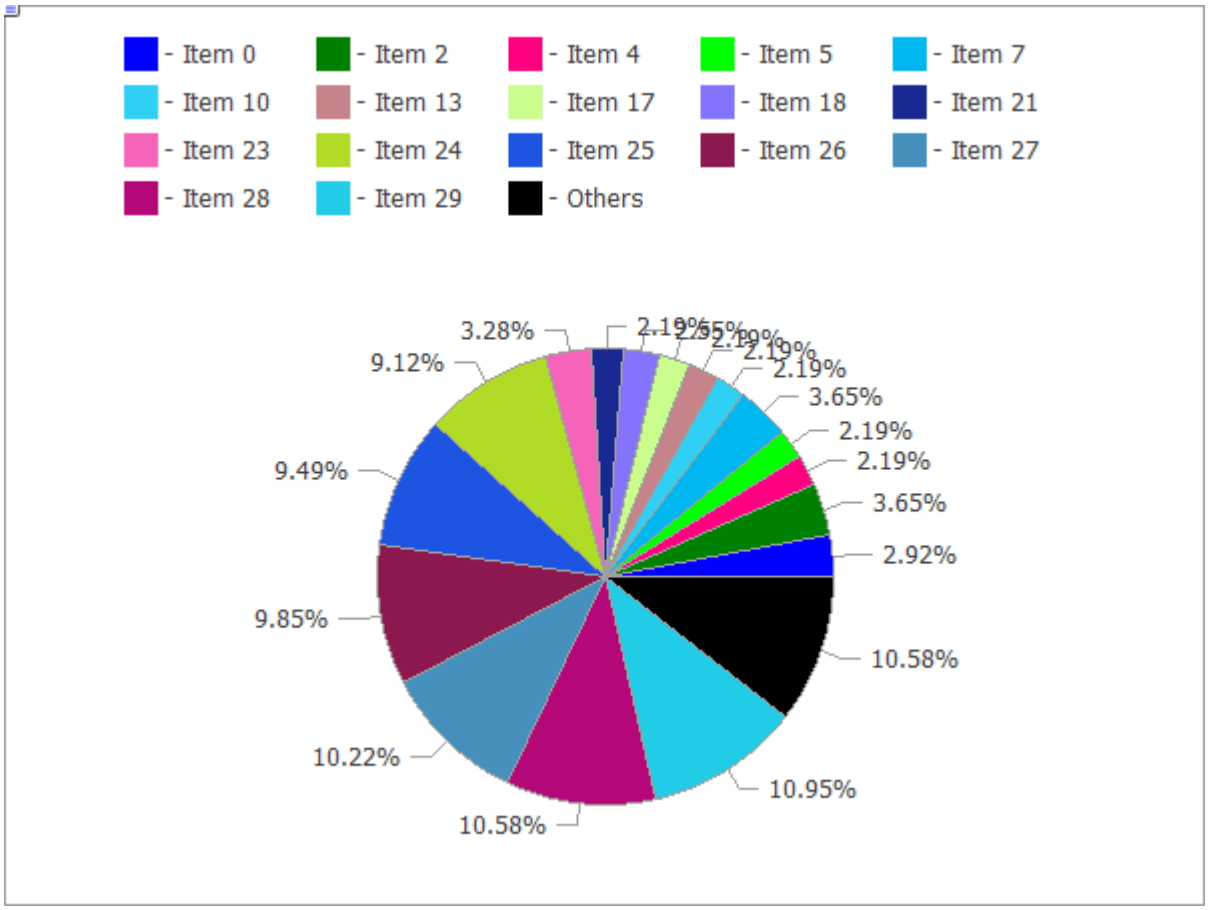
[in] Serinin indisi (serilerin numaralandırılmasına 0 ile başlanır).

Dönüş Değeri

Veri setine göre çizilen eğrinin altında kalan alan.

CPieChart

Pay grafikleri için tasarlanmıştır.



Yukarıdaki şekli oluşturmak için kullanılan kod aşağıda paylaşılmıştır.

Açıklama

Bu sınıfta yer alan yöntemler, grafiksel kaynağın oluşturulmasından, dilim etiketlerinin düzenlenmesine kadar, pay grafiklerle yapılabilecek her türlü kapsamlı çalışmalar için tasarlanmıştır.

Bildirim

```
class CPieChart : public CChartCanvas
```

Başlık

```
#include <Canvas\Charts\PieChart.mqh>
```

Kalıtım hiyerarşisi

CCanvas

CChartCanvas

CPieChart

Sınıf yöntemleri

Yöntem	Eylem
Create	Grafiksel kaynağı oluşturan sanal yöntem.
SeriesSet	Pay grafikte görüntülemek için bir değerler serisi ayarlar.
ValueAdd	Pay grafiğin sonuna yeni bir değer ekler.
ValueInsert	Pay grafiğin belirtilen konumuna yeni bir değer ekler.
ValueUpdate	Pay grafiğin belirtilen konumundaki veriyi günceller.
ValueDelete	Pay grafiğin belirtilen konumundaki veriyi siler.
DrawChart	Tüm bileşenleriyle birlikte bir pay grafik çizmek için tasarlanmış sanal yöntem.
DrawPie	Belirtilen değere karşılık gelen grafik dilimini çizer
LabelMake	Dilimin değerine ve orjinal etiket değerine göre bir dilim etiketi ayarlar.

CCanvas sınıfından türetilen yöntemler

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

CChartCanvas sınıfından türetilen yöntemler

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

Örnek

```
//+------------------------------------------------------------------+
//|                                                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 "Pay grafik kullanım ırneği"
//---
#include <Canvas\Charts\PieChart.mqh>
//+------------------------------------------------------------------+
//| girdiler                                                         |
//+------------------------------------------------------------------+
input int      Width=600;
input int      Height=450;
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
int OnStart(void)
  {
//--- kontrol et
   if(Width<=0 || Height<=0)
     {
     Print("Fazla basit.");
      return(-1);
     }
//--- ızelge oluştur
   CPieChart pie_chart;
   if(!pie_chart.CreateBitmapLabel("PieChart",10,10,Width,Height))
     {
     Print("Hata, pay grafik oluşturulamadı: ",GetLastError());
      return(-1);
     }
   pie_chart.ShowPercent();
//--- ız
   for(uint i=0;i<30;i++)
     {
      pie_chart.ValueAdd(100*(i+1),"Item "+IntegerToString(i));
      Sleep(10);
     }
   Sleep(2000);
//--- başlıkları devre dışı bırak
   pie_chart.LegendAlignment(ALIGNMENT_LEFT);
   Sleep(2000);
//--- başlıkları devre dışı bırak
   pie_chart.LegendAlignment(ALIGNMENT_RIGHT);
   Sleep(2000);
//--- başlıkları devre dışı bırak
   pie_chart.LegendAlignment(ALIGNMENT_TOP);
```

```
   Sleep(2000);
//--- başlıkları devre dışı bırak
   pie_chart.ShowLegend(false);
   Sleep(2000);
//--- yьzdeyi gцsterme
   pie_chart.ShowPercent(false);
   Sleep(2000);
//--- asıklamaları kapat
   pie_chart.ShowDescriptors(false);
   Sleep(2000);
//--- hepsini gцster
   pie_chart.ShowLegend();
   pie_chart.ShowValue();
   pie_chart.ShowDescriptors();
   Sleep(2000);
//--- veya bu şekilde
   pie_chart.ShowFlags(FLAG_SHOW_LEGEND|FLAG_SHOW_DESCRIPTORS|FLAG_SHOW_PERCENT);
   uint total=pie_chart.DataTotal();
//--- değerlerle oyna
   for(uint i=0;i<total && !IsStopped();i++)
     {
      pie_chart.ValueUpdate(i,100*(rand()%10+1));
      Sleep(1000);
     }
//--- renklerle oyna
   for(uint i=0;i<total && !IsStopped();i++)
     {
      pie_chart.ColorUpdate(i%total,RandomRGB());
      Sleep(1000);
     }
//--- dцndьr
   while(!IsStopped())
     {
      pie_chart.DataOffset(pie_chart.DataOffset()+1);
      Sleep(200);
     }
//--- bitir
   pie_chart.Destroy();
   return(0);
  }
//+------------------------------------------------------------------+
//| Rassal RGB değeri                                                |
//+------------------------------------------------------------------+
uint RandomRGB(void)
  {
   return(XRGB(rand()%255,rand()%255,rand()%255));
  }
```

Create

Grafiksel kaynağı oluşturan sanal yöntem.

```
virtual bool  Create(
  const string       name,    // isim
  const int          width,   // genişlik
  const int          height,  // yükseklik
  ENUM_COLOR_FORMAT  clrfmt,  // renk biçimi
  )
```

Parametreler

name

[in] Grafiksel kaynak ismi için temel kelime. Kaynak ismi, oluşturma işlemi sırasında kök kelimeye pseudo-rassal bir dizgi eklenerek meydana getirilir.

width

[in] Piksel bazında genişlik (X-ekseni üzerindeki boyut).

height

[in] Piksel bazında yükseklik (Y-ekseni üzerindeki boyut).

clrfmt

[in] Renk işleme yöntemi. Renk işleme yöntemleri hakkında daha fazla bilgi için ResourceCreate() fonksiyonunun açıklamasına bakın.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

SeriesSet

Pay grafikte görüntülemek için bir değerler serisi ayarlar.

```
bool  SeriesSet(
  const double&  value[],  // değerler
  const string&  text[],   // etiketler
  const uint&    clr[],    // renk
  )
```

Parametreler

value[]

[in] Değerler dizisi.

text[]

[in] Değer etiketlerinin dizisi.

clr[]

[in] Değerlerin renklerinin dizisi.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

ValueAdd

Pay grafiğin sonuna yeni bir değer ekler.

```
bool  ValueAdd(
  const double  value,  // değer
  const string  descr,  // etiket
  const uint    clr,    // renk
  )
```

Parametreler

value

[in] Değer.

descr

[in] Değer etiketi.

clr

[in] Değer rengi.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

ValueInsert

Pay grafiğin belirtilen konumuna yeni bir değer ekler.

```
bool  ValueInsert(
  const uint    pos,    // indis
  const double  value,  // değer
  const string  descr,  // etiket
  const uint    clr,    // renk
  )
```

Parametreler

pos

[in] Giriş indisi.

value

[in] Değer.

descr

[in] Değer etiketi.

clr

[in] Değer rengi.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

ValueUpdate

Pay grafiğin belirtilen konumundaki veriyi günceller.

```
bool  ValueUpdate(
  const uint    pos,    // indis
  const double  value,  // değer
  const string  descr,  // etiket
  const uint    clr,    // renk
  )
```

Parametreler

pos

[in] Verinin indisi (serilerin numaralandırılmasına 0 ile başlanır).

value

[in] Değer.

descr

[in] Değer etiketi.

clr

[in] Değer rengi.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

ValueDelete

Pay grafiğin belirtilen konumundaki veriyi siler.

```
bool  ValueDelete(
  const uint  pos,  // indis
  )
```

Parametreler

pos

[in] Verinin indisi (serilerin numaralandırılmasına 0 ile başlanır).

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'.

DrawChart

Tüm bileşenleriyle birlikte bir pay grafik çizmek için tasarlanmış sanal yöntem.

```
virtual void  DrawChart()
```

DrawPie

Belirtilen değere karşılık gelen grafik dilimini çizer

```
void  DrawPie(
  double       fi3,   // dilim merkezinden gelen ilk ışının açısı
  double       fi4,   // dilim merkezinden gelen ikinci ışının açısı
  int          idx,   // indis
  CPoint&      p[],  //
  const uint   clr,   //
  )
```

Parametreler

fi3

[in] Yayın ilk sınırını belirleyen açının radyan cinsinden değeri.

fi4

[in] Yayın ikinci sınırını belirleyen açının radyan cinsinden değeri.

idx

[in] Dilime karşılık gelen indis değeri.

p[]

[in] Dilimin çizimi için referans noktaları (x, y) dizisi.

clr

[in] Dilim rengi.

LabelMake

Dilimin değerine ve orjinal etiket değerine göre bir dilim etiketi ayarlar.

```
string  LabelMake(
  const string  text,     // etiket
  const double  value,    // değer
  const bool    to_left,  // bayrak
  )
```

Parametreler

text

[in] Etiket.

value

[in] Değer.

to_left

[in] Etiketlerin dizilişini belirtir:

- true — değer etiketten sonra gelir.
- false — değer etiketten önce gelir.

Dönüş Değeri

Dilim etiketi.

3D grafik

Bu bölüm, üç boyutlu grafik geliştirme için sınıflar içerir. Sınıflar DirectX ile çalışma fonksiyonlarına dayanır. CCnavas3D; dokular, gölgelendiriciler, köşe tamponları, indeksler ve gölgelendirici parametreleri gibi grafik kaynaklarının yöneticisine sahip olan ve ayrıca kamera ve ışıklandırmayı yönetme metodlarını içeren bir temel sınıftır.

Kütüphaneyle çalışmaya başlamak için MetaTrader 5'te DirectX'i kullanarak 3D grafik oluşturma makalesini okuyun.

Ayrıca; kutu, kullanıcı verilerine ilişkin üç boyutlu yüzey ve rassal ızgara gibi sahne tabanı nesnelerinin sınıfları da vardır.

CCanvas3D

CCanvas3D, fiyat grafiği üzerinde 3D nesnelerin basitleştirilmiş şekilde oluşturulması ve görselleştirilmesi için bir sınıftır.

Açıklama

CCanvas3D, büyük miktarlarda verilerin animasyonlu 3D grafikler biçiminde oluşturulmasını ve görselleştirilmesini büyük ölçüde basitleştirir. Sınıf; dokular, gölgelendiriciler, köşe tamponları, indeksler ve gölgelendirici parametreleri gibi grafik kaynaklarının yöneticisini ve ayrıca kamera ve ışıklandırmayı yönetme metodlarını içerir.

Ayrıca kütüphane; kutu, kullanıcı verilerine ilişkin üç boyutlu yüzey ve rassal ızgara gibi sahne tabanı nesnelerinin sınıflarını da içerir.

Grafik kartı, fonksiyonların çalışması için DX 11 ve Shader Model 5.0'ı desteklemelidir.

Bildirme

```
class CCanvas
```

Başlık

```
#include <Canvas\Canvas.mqh>
```

Kalıtım hiyerarşisi

CCanvas

CCanvas3D

Gruplara göre sınıf metodları

Oluşturma/silme	Açıklama
Create	Bir fiyat grafiği nesnesine bağlı kalmadan 3D sahneyi render almak bir grafik kaynağı oluşturur.
Attach	OBJ_BITMAP_LABEL nesnesinden bir grafik kaynağı elde eder ve bunu CCanvas sınıfının bir örneğine ekler.
ObjectAdd	Sonraki render işlemi için 3D sahneye bir nesne ekler.
Destroy	Grafik kaynağını yok eder ve bir 3D grafik içeriği serbest bırakır.
Işıklandırma	
AmbientColorSet	Çok yönlü ortam ışıklandırmasının rengini ve yoğunluğunu ayarlar.
AmbientColorGet	Çok yönlü ortam ışıklandırmasının rengini ve yoğunluğunu elde eder.
LightDirectionSet	Yönlü bir ışık kaynağının yönünü ayarlar.

Oluşturma/silme	Açıklama
LightDirectionGet	Yönlü bir ışık kaynağının yönünü elde eder.
LightColorSet	Yönlü bir ışık kaynağının rengini ve yoğunluğunu ayarlar.
LightColorGet	Yönlü bir ışık kaynağının rengini ve yoğunluğunu elde eder.
Kamera	
ProjectionMatrixSet	3D koordinatın izdüşüm matrisini 2D kareye hesaplar ve ayarlar.
ProjectionMatrixGet	3D sahnenin izdüşüm matrisini 2D kare halinde elde eder.
ViewMatrixSet	3D sahnenin görünüm matrisini ayarlar.
ViewMatrixGet	3D sahnenin görünüm matrisini geri döndürür.
ViewPositionSet	3D sahneye bir bakış noktası ayarlar.
ViewRotationSet	3D sahnede bakış yönünü ayarlar.
ViewTargetSet	Bakışın yönlendirildiği noktanın koordinatlarını ayarlar.
ViewUpDirectionSet	3D alanda karenin üst kenarının yönünü ayarlar.
Render alma	
Render	Sonraki görüntüleme için kare iç arabelleğindeki tüm sahne nesnelerini render alır.
RenderBegin	Yeni bir kare render almak için bir grafik içeriği hazırlar.
RenderEnd	Oluşturulan kareyi iç arabelleğe kopyalar ve gerekirse fiyat grafiğindeki görüntüyü günceller.
Kaynakları elde etme	
DXContext	Grafik içeriği tanıtıcı değerini elde eder.
DXDispatcher	Kaynak yöneticisi tanıtıcı değerini elde eder.
InputScene	Sahne parametreleri arabelleğine olan işaretçiyi elde eder.

AmbientColorGet

Çok yönlü ortam ışıklandırmasının rengini ve yoğunluğunu elde eder.

```
void  AmbientColorGet(
   DXColor  &ambient_color      // çok yönlü ışıklandırmanın rengi ve yoğunluğu
   );
```

Parametreler

&ambient_color

[out] Çok yönlü ışıklandırmanın rengi.

Geri dönüş değeri

Yok.

Not

Yoğunluk, DXColor yapısının alfa kanalında saklanır.

AmbientColorSet

Çok yönlü ortam ışıklandırmasının rengini ve yoğunluğunu ayarlar.

```
void  AmbientColorSet(
   const DXColor  &ambient_color      // çok yönlü ışıklandırmanın rengi ve yoğunluğu
   );
```

Parametreler

&ambient_color

[in] Çok yönlü ışıklandırmanın rengi.

Geri dönüş değeri

Yok.

Not

Yoğunluk, DXColor yapısının alfa kanalında ayarlanır.

Attach

OBJ_BITMAP_LABEL nesnesinden bir grafik kaynağı elde eder ve bunu CCanvas sınıfının bir örneğine ekler.

```
bool  Attach(
   const long          chart_id,                          // fiyat grafiği ID'si
   const string        objname,                           // nesne adı
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // renk işleme metodu
   )
```

OBJ_BITMAP_LABEL nesnesi için bir grafik kaynağı oluşturur ve bunu CCanvas sınıfının bir örneğine ekler.

```
bool  Attach(
   const long          chart_id,                          // fiyat grafiği ID'si
   const string        objname,                           // nesne adı
   const int           width,                             // piksel cinsinden görür
   const int           height,                            // piksel cinsinden görür
   ENUM_COLOR_FORMAT   clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // renk işleme metodu
   )
```

Parametreler

chart_id

[in] Fiyat grafiği ID'si.

objname

[in] Grafik nesnesinin adı.

width

[in] Kaynaktaki karenin genişliği.

height

[in] Kare yüksekliği.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Renk işleme metodu. Renk işleme yöntemleri hakkında daha fazla bilgi için ResourceCreate() fonksiyonunun açıklamasına bakın.

Not

başarılıysa - true, grafik nesnesi eklenemediğinde - false.

Create

Bir fiyat grafiği nesnesine bağlı kalmadan 3D sahneyi render almak bir grafik kaynağı oluşturur.

```
virtual bool  Create(
   const string       name,                              // grafik nesnesinin adı
   const int          width,                             // genişlik
   const int          height,                            // yükseklik
   ENUM_COLOR_FORMAT  clrfmt=COLOR_FORMAT_XRGB_NOALPHA   // renk formatı
   );
```

Parametreler

name

[in] Grafik nesnesinin adı

width

[in] Kare genişliği.

height

[in] Kare yüksekliği.

clrfmt=COLOR_FORMAT_XRGB_NOALPHA

[in] Renk işleme metodu. Renk işleme yöntemleri hakkında daha fazla bilgi için ResourceCreate() fonksiyonunun açıklamasına bakın.

Not

bir kaynak oluşturulursa - true, aksi takdirde - false.

Destroy

Grafik kaynağını yok eder ve bir 3D grafik içeriği serbest bırakır.

```
virtual void  Destroy()
```

Geri dönüş değeri

Yok.

Not

Grafik kaynağı bir fiyat grafiği nesnesine bağlı kalınarak oluşturulduysa, fiyat grafiği nesnesi silinir.

DXContext

Grafik içeriği tanıtıcı değerini elde eder.

```
int  DXContext()
```

Geri dönüş değeri

Grafik içeriği tanıtıcı değeri.

DXDispatcher

Kaynak yöneticisi tanıtıcı değerini elde eder.

```
CDXDispatcher*  DXDispatcher()
```

Geri dönüş değeri

Kaynak yöneticisi tanıtıcı değeri.

InputScene

Sahne parametreleri arabelleğine olan işaretçiyi elde eder.

```
CDXInput*  InputScene()
```

Geri dönüş değeri

Sahne parametreleri arabelleğine olan işaretçi.

LightColorGet

Yönlü bir ışık kaynağının rengini ve yoğunluğunu elde eder.

```
void  LightColorGet(
   DXColor  &light_color      // yönlü ışık kaynağının rengi ve yoğunluğu
   );
```

Parametreler

&light_color

[out] Yönlü ışık kaynağının rengi ve yoğunluğu.

Geri dönüş değeri

Yok.

Not

Yoğunluk, DXColor yapısının alfa kanalında saklanır.

LightColorSet

Yönlü bir ışık kaynağının rengini ve yoğunluğunu ayarlar.

```
void  LightColorSet(
   const DXColor  &light_color      // yönlü ışık kaynağının rengi ve yoğunluğu
   );
```

Parametreler

&light_color

[in] Yönlü ışık kaynağının rengi ve yoğunluğu.

Geri dönüş değeri

Yok.

Not

Yoğunluk, DXColor yapısının alfa kanalında ayarlanır.

LightDirectionGet

Yönlü bir ışık kaynağının yönünü elde eder.

```
void  LightDirectionGet(
   DXVector3  &light_direction      // yön vektörü
   );
```

Parametreler

&light_direction

[out] Yön vektörü.

Geri dönüş değeri

Yok.

LightDirectionSet

Yönlü bir ışık kaynağının yönünü ayarlar.

```
void  LightDirectionSet(
   const DXVector3  &light_direction      // yön vektörü
   );
```

Parametreler

&light_direction

[in] Yön vektörü.

Geri dönüş değeri

Yok.

ObjectAdd

Sonraki render işlemi için 3D sahneye bir nesne ekler.

```
bool  ObjectAdd(
   CDXObject  *object      // nesneye olan işaretçi
   );
```

Parametreler

object

[in] CDXObject soyut sınıfından türetilmiş sınıfın bir örneğine olan işaretçi.

Geri dönüş değeri

başarılıysa - true, 3D grafik nesnesi eklenemediğinde - false.

ProjectionMatrixGet

3D sahnenin izdüşüm matrisini 2D kare halinde elde eder.

```
void  ProjectionMatrixGet(
   DXMatrix  &projection_matrix      // izdüşüm matrisi
   );
```

Parametreler

&projection_matrix

[out] İzdüşüm matrisi.

Geri dönüş değeri

Yok.

ProjectionMatrixSet

3D koordinatın izdüşüm matrisini 2D kareye hesaplar ve ayarlar.

```
void  ProjectionMatrixSet(
   float  fov,              // görüş alanı
   float  aspect_ratio,     // kare en-boy oranı
   float  z_near,           //
   float  z_far             //
   );
```

Parametreler

fov

[in] Bir sahne izdüşümü oluşturmak için radyan cinsinden görüş alanı genişliği.

aspect_ratio

[in] 2D kare en-boy oranı.

z_near

[in] Yakın kırpma düzlemine olan uzaklık.

z_far

[in] Uzak kırpma düzlemine olan uzaklık.

Geri dönüş değeri

Yok.

Not

2D kare yalnızca belirtilen görüş alanına düşen ve yakın ve uzak kırpma düzlemleri arasında bulunan 3D nesnelerin izdüşümlerini görüntüler.

Render

Sonraki görüntüleme için kare iç arabelleğindeki tüm sahne nesnelerini render alır.

```
bool  Render(
   uint  flags,                  // bayrak kombinasyonu
   uint  background_color=0      // arka plan rengi
   );
```

Parametreler

flags

[in] Render alma modunu ayarlayan bayrakların kombinasyonu. Olası değerler:
DX_CLEAR_COLOR - *background_color*'ı kullanarak görüntü arabelleğini temizle.
DX_CLEAR_DEPTH - derinlik arabelleğini temizle.

background_color=0

[in] 3D sahnenin arka plan rengi.

Geri dönüş değeri

başarılıysa - true, render alınamıyorsa - false.

Not

Render()'ı çağırmak, fiyat grafiğindeki sahneyi güncellemez. Bunun yerine, yalnızca görüntünün iç arabelleğini günceller. Güncellenmiş kareyi render almak için Update() metodu açıkça çağrılmalıdır.

Render(), RenderBegin ve RenderEnd() çağrılarını içerir.

RenderBegin

Yeni bir kare render almak için bir grafik içeriği hazırlar.

```
virtual bool  RenderBegin(
   uint  flags,                  // bayrak kombinasyonu
   uint  background_color=0      // arka plan rengi
   );
```

Parametreler

flags

[in] Render alma modunu ayarlayan bayrakların kombinasyonu. Olası değerler:
DX_CLEAR_COLOR - *background_color*'ı kullanarak görüntü arabelleğini temizle.
DX_CLEAR_DEPTH - derinlik arabelleğini temizle.

background_color=0

[in] 3D sahnenin arka plan rengi.

Geri dönüş değeri

başarılıysa - true, gölgelendirici girdilerini güncelleştiremezse - false.

RenderEnd

Oluşturulan kareyi iç arabelleğe kopyalar ve gerekirse fiyat grafiğindeki görüntüyü günceller.

```
virtual bool  RenderEnd(
   bool  redraw=false      // güncelleme bayrağı
   );
```

Parametreler

redraw=false

[in] Fiyat grafiğinin yeniden çizilmesi gerektiğini belirten bayrak.

Geri dönüş değeri

başarılıysa - true, aksi takdirde - false.

ViewMatrixGet

3D sahnenin görünüm matrisini geri döndürür.

```
void  ViewMatrixGet(
   DXMatrix  &view_matrix      // görünüm matrisi
   );
```

Parametreler

&view_matrix

[out] 3D alanda kamera konumunu ve yönünü ayarlayan görünüm matrisi.

Geri dönüş değeri

Yok.

ViewMatrixSet

3D sahnenin görünüm matrisini ayarlar.

```
void  ViewMatrixSet(
   const DXMatrix  &view_matrix      // görünüm matrisi
   );
```

Parametreler

&view_matrix

[in] 3D alanda kamera konumunu ve yönünü ayarlayan görünüm matrisi.

Geri dönüş değeri

Yok.

ViewPositionSet

3D sahneye bir bakış noktası ayarlar.

```
void  ViewPositionSet(
   const DXVector3  &position      // bakış noktasının konumu
   );
```

Parametreler

&position

[in] 3D sahnede bakış noktasının konumu.

Geri dönüş değeri

Yok.

Not

ViewPositionSet()'i kullanarak bir bakış noktası konumu ayarlamak, ViewMatrixGet()'te elde edilen görünüm matrisini değiştirir.

ViewRotationSet

3D sahnede bakış yönünü ayarlar.

```
void  ViewRotationSet(
   const DXVector3  &rotation      // döndürme açısı vektörü
   );
```

Parametreler

&rotation

[in] 3D sahnede bakış yönünü hesaplamak için Euler açılarını belirten vektör.

Geri dönüş değeri

Yok.

Not

ViewRotationSet()'i kullanarak bakış yönünü ayarlamak, ViewMatrixGet()'te elde edilen görünüm matrisini değiştirir.

ViewTargetSet

Bakışın yönlendirildiği noktanın koordinatlarını ayarlar.

```
void  ViewTargetSet(
   const DXVector3  &target      // hedef koordinatlar
   );
```

Parametreler

&target

[in] Bakışın yönlendirildiği noktanın koordinatları

Geri dönüş değeri

Yok.

Not

Bakış noktasını hareket ettirirken bakışı bir sahne noktasına sabitlemek için kullanılır.

ViewRotationSet()'i kullanarak yeni bir hedef koordinatı ayarlamak, ViewMatrixGet()'te elde edilen görünüm matrisini değiştirir.

ViewTargetSet(), bakış yönünü tanımlamak için ViewUpDirectionSet() ile birlikte kullanılır.

ViewUpDirectionSet

3D alanda karenin üst kenarının yönünü ayarlar.

```
void  ViewUpDirectionSet(
   const DXVector3  &up_direction      // üst yön
   );
```

Parametreler

&up_direction

[in] 3D alanda karenin üst kısmının yönü.

Geri dönüş değeri

Yok.

Not

ViewUpDirectionSet()'i kullanarak yeni bir yön ayarlamak, ViewMatrixGet()'te elde edilen görünüm matrisini değiştirir.

ViewUpDirectionSet(), bakış yönünü tanımlamak için ViewTargetSet() ile birlikte kullanılır.

CChart

CChart sınıfı, "Çizelge" grafik nesnesinin özelliklerine kolay erişim için düzenlenmiş bir sınıftır.

Açıklama

CChart "Çizelge" grafik nesnesinin özelliklerine kolay erişim sağlar.

Bildirim

```
class CChart : public CObject
```

Başlık

```
#include <Charts\Chart.mqh>
```

Kalıtım hiyerarşisi

CObject

CChart

Sınıf Yöntemleri

Korunan veriye erişim	
ChartID	Çizelge tanımlayıcısını alır
Genel özellikler	
Mode	"Mod" özelliğinin değerini alır/ayarlar (grafik modu: çubuk, mum veya çizgi)
Foreground	"Ön-plan" özelliğinin değerini alır/ayarlar
Shift	"Kaydırma" (taşıma) özelliğinin değerini alır/ayarlar
ShiftSize	"Kaydırma boyutu" değerini (yüzdelik olarak) alır/ayarlar
AutoScroll	"Otomatik kaydır" özelliğinin değerini alır/ayarlar
Scale	"Ölçek" özelliğinin değerini alır/ayarlar
ScaleFix	"Sabit ölçek" özelliğinin değerini alır/ayarlar (sabit veya sabit değil)
ScaleFix_11	"Birebir sabit ölçek" özelliğinin değerini alır/ayarlar (1:1 veya değil)
FixedMax	"Sabit maksimum" özelliğinin (sabit maksimum fiyat) değerini alır/ayarlar
FixedMin	"Sabit minimum" özelliğinin (sabit minimum fiyat) değerini alır/ayarlar
ScalePPB	"Çubuk başı ölçek puanı" özelliğinin değerini alır/ayarlar (çubuk başı puan veya değil)

Korunan veriye erişim	
PointsPerBar	"Çubuk başı puan" özelliğinin değerini alır/ayarlar (çubuk başına düşen puan bazında)
Gösterim özellikleri	
ShowOHLC	"OHLC göster" özelliğinin değerini alır/ayarlar
ShowLineBid	"Satış fiyat çizgisini göster" özelliğinin değerini alır/ayarlar
ShowLineAsk	"Alış fiyat çizgisini göster" özelliğinin değerini alır/ayarlar
ShowLastLine	"Son fiyat çizgisini göster" özelliğinin değerini alır/ayarlar
ShowPeriodSep	"Periyot ayraçlarını göster" özelliğinin değerini alır/ayarlar
ShowGrid	"Izgarayı göster" özelliğinin değerini alır/ayarlar
ShowVolumes	"Hacimleri göster" özelliğinin değerini alır/ayarlar
ShowObjectDescr	"Nesne açıklamalarını göster" özelliğinin değerini alır/ayarlar
ShowDateScale	"Zaman ölçeğini göster" özelliğinin değerini ayarlar (çizelgenin zaman ölçeği)
ShowPriceScale	"Fiyat ölçeğini göster" özelliğinin değerini ayarlar (çizelgenin fiyat ölçeği)
Renk özellikleri	
ColorBackground	"Arka-plan rengi" özelliğinin değerini alır/ayarlar
ColorForeground	"Ön-plan rengi" özelliğinin değerini alır/ayarlar (Çizelgenin eksenleri, ölçeği ve OHLC dizgilerinin rengi)
ColorGrid	"Izgara rengi" özelliğinin değerini alır/ayarlar
ColorBarUp	"Yükseliş çubuğu rengi" özelliğinin değerini alır/ayarlar (yukarı yönlü çubukların, gölgelerin ve mum gövde çizgilerinin rengi)
ColorBarDown	"Düşüş çubuğu rengi" özelliğinin değerini alır/ayarlar (aşağı yönlü çubukların, gölgelerin ve mum gövde çizgilerinin rengi)
ColorCandleBull	"Yükseliş mumu rengi" özelliğinin değerini alır/ayarlar (yukarı yönlü mumların gövde rengi)
ColorCandleBear	"Düşüş mumu rengi" özelliğinin değerini alır/ayarlar (aşağı yönlü mumların gövde rengi)
ColorChartLine	"Çizgi grafiği rengi" özelliğinin değerini alır/ayarlar (çizgi grafiği ve Doji mumlarının rengi)
ColorVolumes	"Hacim rengi" özelliğinin değerini alır/ayarlar (hacimlerin ve açık pozisyonların rengi)
ColorLineBid	"Satış fiyatı çizgisi rengi" özelliğinin değerini alır/ayarlar

Korunan veriye erişim	
ColorLineAsk	"Alış fiyatı çizgisi rengi" özelliğinin değerini alır/ayarlar
ColorLineLast	"Son fiyat çizgisi rengi" özelliğinin değerini alır/ayarlar
ColorStopLevels	"Stop seviyeleri rengi" özelliğinin değerini alır/ayarlar
Salt Okunur özellikler	
VisibleBars	Görünür çizelge çubuklarının toplam sayısını alır
WindowsTotal	Çizelge pencerelerinin (alt-pencereler dahil) toplam sayısını alır
WindowIsVisible	Belirtilen alt-pencerenin görünürlük bayrağını alır
WindowHandle	Çizelgenin 'pencere tanıtıcı değerini' alır (HWND)
FirstVisibleBar	Çizelgenin görünen ilk çubuğunun numarasını alır
WidthInBars	Çubuk sayısı bazında pencere genişliğini alır.
WidthInPixels	Piksel bazında alt-pencere genişliğini alır.
HeightInPixels	Piksel bazında alt-pencere yüksekliğini alır.
PriceMin	Belirtilen alt penceredeki en küçük fiyat değerini alır
PriceMax	Belirtilen alt penceredeki en büyük fiyat değerini alır
Özellikler	
Attach	Mevcut çizelgeyi sınıf örneğine tutturur
FirstChart	Müşteri terminalindeki ilk çizelgeyi sınıf örneğine tutturur
NextChart	Müşteri terminalindeki bir sonraki çizelgeyi sınıf örneğine tutturur
Open	Belli parametrelerle bir çizelge açar ve onu sınıf örneğine tutturur
Detach	Çizelgeyi sınıf örneğinden ayırır
Close	Sınıf örneğine tutturulmuş çizelgeyi kapatır
BringToTop	Çizelgeyi diğer çizelgelerin üzerine taşır
EventObjectCreate	Çizelge üzerinde yeni nesne oluşturma olayı hakkında uyarı göndermek için bir bayrak ayarlar
EventObjectDelete	Çizelge üzerindeki bir nesnenin silinme olayı hakkında uyarı göndermek için bir bayrak ayarlar
Göstergeler	
IndicatorAdd	Belirtilen tanıtıcı değere sahip bir göstergeyi belirtilen çizelge alt-penceresine ekler

Korunan veriye erişim	
IndicatorDelete	Belirtilen tanıtıcı değere sahip bir göstergeyi belirtilen çizelge alt-penceresinden kaldırır
IndicatorsTotal	Belli bir alt-pencereye uygulanmış tüm göstergelerin sayısını verir
IndicatorName	Belli bir alt-penceredeki göstergenin kısa ismini verir
Gezinti	
Navigate	Çizelgeyi kaydırır
MQL5 API erişimi	
Symbol	Çizelge sembolünü alır
Period	Çizelge periyodunu alır
Redraw	Sınıf örneğine tutturulan çizelgeyi yeniden çizer
GetInteger	Bu fonksiyon karşılık gelen nesne özelliğinin değerine dönüş yapar
SetInteger	Tamsayı tipli bir özellik için yeni değer ayarlar
GetDouble	Bu fonksiyon karşılık gelen nesne özelliğinin değerine dönüş yapar
SetDouble	double tipli bir özellik için yeni değer ayarlar
GetString	Bu fonksiyon karşılık gelen nesne özelliğinin değerine dönüş yapar
SetString	string tipli bir özellik için yeni değer ayarlar
SetSymbolPeriod	Sınıf örneğine tutturulan çizelgenin sembol ve periyot değerlerini değiştirir
ApplyTemplate	Belirtilen şablonu çizelgeye uygular
ScreenShot	Belirtilen çizelgenin ekran görüntüsünü alır ve bir .gif dosyasına kaydeder
WindowOnDropped	Nesnenin (Uzman veya Script) bırakma noktasına karşılık gelen alt-pencerenin numarasını alır
PriceOnDropped	Nesnenin (Uzman veya Script) bırakma noktasına karşılık gelen fiyat koordinatını alır
TimeOnDropped	Nesnenin (Uzman veya Script) bırakma noktasına karşılık gelen zaman koordinatını alır
XOnDropped	Nesnenin (Uzman veya Script) bırakma noktasına karşılık gelen X koordinatını alır
YOnDropped	Nesnenin (Uzman veya Script) bırakma noktasına karşılık gelen Y koordinatını alır

Korunan veriye erişim	
Giriş/Çıkış	
virtual Save	Nesne parametrelerini dosyaya kaydeder
virtual Load	Nesne parametrelerini dosyadan yükler
virtual Type	Grafik nesnesinin tip tanımlayıcısını alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

ChartID

Çizelge tanımlayıcısına dönüş yapar.

```
long  ChartID() const
```

Dönüş değeri

Sınıf örneğine atanmış çizelgenin tanımlayıcısı. Atanmış hiçbir nesne yoksa -1 dönüşü yapar.

Mode (Get Yöntemi)

"Mod" özelliğinin değerini alır (grafik modu: çubuk, mum veya çizgi)

```
ENUM_CHART_MODE  Mode() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Mod" özelliğinin değeri. Atanmış hiçbir nesne yoksa WRONG_VALUE değerine dönüş yapar.

Mode (Set Yöntemi)

"Mod" özelliği için yeni değer ayarlar (çubuk, mum veya çizgi).

```
bool  Mode(
   ENUM_CHART_MODE  mode       // yeni çizelge modu
   )
```

Parametreler

mode

[in] ENUM_CHART_MODE sayımının değerlerinden biri (çubuk, mum veya çizgi).

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Foreground (Get Yöntemi)

"Ön-plan" özelliğinin değerini alır.

```
bool  Foreground() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Ön-plan" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

Foreground (Set Yöntemi)

"Ön-plan" özelliği için yeni değer ayarlar.

```
bool  Foreground(
   bool  foreground      // yeni bayrak değeri
   )
```

Parametreler

foreground

[in] "Ön-plan" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Shift (Get Yöntemi)

"Kaydırma" (taşıma) özelliğinin değerini alır.

```
bool  Shift() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Kaydırma" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

Shift (Set Yöntemi)

"Kaydırma" (taşıma) özelliği için yeni değer ayarlar.

```
bool  Shift(
   bool  shift      // yeni bayrak değeri
   )
```

Parametreler

shift

[in] "Kaydırma" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

ShiftSize (Get Yöntemi)

"Kaydırma boyutu" değerini (yüzdelik olarak) alır.

```
double  ShiftSize() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Kaydırma boyutu" özelliğinin değeri. Atanmış hiçbir nesne yoksa EMPTY_VALUE dönüşü yapar.

ShiftSize (Set Yöntemi)

"Kaydırma boyutu" değerini (yüzdelik olarak) yeniden ayarlar.

```
bool  ShiftSize(
   double  shift_size      // yeni özellik değeri
   )
```

Parametreler

shift_size

[in] "Kaydırma boyutu" için yeni değer (yüzdelik olarak).

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

AutoScroll (Get Yöntemi)

"Otomatik kaydır" özelliğinin değerini alır.

```
bool  AutoScroll() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Otomatik kaydır" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

AutoScroll (Set Yöntemi)

"Otomatik kaydır" özelliği için yeni değer ayarlar.

```
bool  AutoScroll(
   bool  autoscroll      // Yeni bayrak değeri
   )
```

Parametreler

autoscroll

[in] "Otomatik kaydır" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

Scale (Get Yöntemi)

"Ölçek" özelliğinin değerini alır.

```
int  Scale() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Ölçek" özelliğinin değeri. Tutturulmuş hiçbir nesne yoksa 0 dönüşü yapar.

Scale (Set Yöntemi)

"Ölçek" özelliği için yeni değer ayarlar.

```
bool  Scale(
   int  scale      // yeni değer
   )
```

Parametreler

scale

[in] "Ölçek" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

ScaleFix (Get Yöntemi)

"Sabit ölçek" özelliğinin değerini alır (sabit veya sabit değil).

```
bool  ScaleFix() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Sabit ölçek" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

ScaleFix (Set Yöntemi)

"Sabit ölçek" özelliği için yeni değer.

```
bool  ScaleFix(
   bool  scale_fix      // yeni değer
   )
```

Parametreler

scale_fix

[in] "Sabit ölçek" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

ScaleFix_11 (Get Yöntemi)

"Birebir sabit ölçek" özelliğinin değerini alır (1:1 veya değil).

```
bool  ScaleFix_11() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Birebir sabit ölçek" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

ScaleFix_11 (Set Yöntemi)

"Birebir sabit ölçek" özelliği için yeni değer ayarlar.

```
bool  ScaleFix_11(
   string  scale_11      // yeni değer
   )
```

Parametreler

scale_11

[in] "Birebir sabit ölçek" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

FixedMax (Get Yöntemi)

"Sabit maksimum" özelliğinin (sabit maksimum fiyat) değerini alır.

```
double  FixedMax() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Sabit maksimum" özelliğinin değeri. Atanmış hiçbir nesne yoksa EMPTY_VALUE dönüşü yapar.

FixedMax (Set Yöntemi)

"Sabit maksimum" özelliği için yeni değer ayarlar.

```
bool  FixedMax(
   double  max       // yeni sabit maksimum
   )
```

Parametreler

max

[in] "Sabit maksimum" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

FixedMin (Get Yöntemi)

"Sabit minimum" özelliğinin (sabit minimum fiyat) değerini alır.

```
double  FixedMin() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Sabit minimum" özelliğinin değeri. Atanmış hiçbir nesne yoksa EMPTY_VALUE dönüşü yapar.

FixedMin (Set Yöntemi)

"Sabit minimum" özelliği için yeni değer ayarlar.

```
bool  FixedMax(
   double  min      // yeni sabit minimum
   )
```

Parametreler

max

[in] "Sabit minimum" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

PointsPerBar (Get Yöntemi)

"Çubuk başı puan" özelliğinin değerini alır (çubuk başına düşen puan bazında).

```
double  PointsPerBar() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Çubuk başı puan" özelliğinin değeri. Atanmış hiçbir nesne yoksa EMPTY_VALUE değerine dönüş yapar.

PointsPerBar (Set Yöntemi)

"Çubuk başı puan" özelliği için yeni değer ayarlar.

```
bool  PointsPerBar(
   double  ppb      // yeni ölçek değeri
   )
```

Parametreler

ppb

[in] Yeni ölçek değeri (çubuk başına düşen puan bazında).

Dönüş değeri

Başarılı ise 'true', ölçek değişmediyse 'false'.

ScalePPB (Get Yöntemi)

"Çubuk başı ölçek puanı" özelliğinin değerini alır (çubuk başı puan veya değil).

```
bool  ScalePPB() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "Çubuk başı ölçek puanı" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

ScalePPB (Set Yöntemi)

"Çubuk başı ölçek puanı" özelliği için yeni değer ayarlar.

```
bool  ScalePPB(
   bool  scale_ppb      // yeni bayrak değeri
   )
```

Parametreler

scale_ppb

[in] "Çubuk başı ölçek puanı" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

ShowOHLC (Get Yöntemi)

"OHLC göster" özelliğinin değerini alır.

```
bool  ShowOHLC() const
```

Dönüş değeri

Sınıf örneğine atanan nesnenin "OHLC göster" özelliğinin değeri. Hiçbir nesne atanmamışsa 'false' dönüşü yapar.

ShowOHLC (Set Yöntemi)

"OHLC göster" özelliği için yeni değer ayarlar.

```
bool  ShowOHLC(
   bool  show      // yeni değer
   )
```

Parametreler

show

[in] "OHLC göster" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

ShowLineBid (Get Yöntemi)

"Satış fiyat çizgisini göster" özelliğinin değerini alır.

```
bool  ShowLineBid() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Satış fiyat çizgisini göster" özelliğinin değeri. Hiçbir çizelge atanmamışsa 'false' dönüşü yapar.

ShowLineBid (Set Yöntemi)

"Satış fiyat çizgisini göster" özelliği için yeni değer ayarlar.

```
bool  ShowLineBid(
   bool  show      // yeni değer
   )
```

Parametreler

show

[in] "Satış fiyat çizgisini göster" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

ShowLineAsk (Get Yöntemi)

"Alış fiyat çizgisini göster" özelliğinin değerini alır.

```
bool  ShowLineAsk() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Alış fiyat çizgisini göster" özelliğinin değeri. Hiçbir çizelge atanmamışsa 'false' dönüşü yapar.

ShowLineAsk (Set Yöntemi)

"Alış fiyat çizgisini göster" özelliği için yeni değer ayarlar.

```
bool  ShowLineAsk(
   bool  show      // yeni değer
   )
```

Parametreler

show

[in] "Alış fiyat çizgisini göster" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

ShowLastLine (Get Yöntemi)

"Son fiyat çizgisini göster" özelliğinin değerini alır.

```
bool  ShowLastLine() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Son fiyat çizgisini göster" özelliğinin değeri. Hiçbir çizelge atanmamışsa 'false' dönüşü yapar.

ShowLastLine (Set Yöntemi)

"Son fiyat çizgisini göster" özelliği için yeni değer ayarlar.

```
bool  ShowLastLine(
   bool  show      // yeni bayrak değeri
   )
```

Parametreler

show

[in] "Son fiyat çizgisini göster" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

ShowPeriodSep (Get Yöntemi)

"Periyot ayraçlarını göster" özelliğinin değerini alır.

```
bool  ShowPeriodSep() const
```

Dönüş değeri

Sınıf örneğine tutturulan çizelgenin "Periyot ayraçlarını göster" özelliğinin değeri. Hiçbir çizelge atanmamışsa 'false' dönüşü yapar.

ShowPeriodSep (Set Yöntemi)

"Periyot ayraçlarını göster" özelliği için yeni değer ayarlar.

```
bool  ShowPeriodSep(
   bool  show      // yeni değer
   )
```

Parametreler

show

[in] "Periyot ayraçlarını göster" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

ShowGrid (Get Yöntemi)

"Izgarayı göster" özelliğinin değerini alır.

```
bool  ShowGrid() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Izgarayı göster" özelliğinin değeri. Hiçbir çizelge atanmamışsa 'false' dönüşü yapar.

ShowGrid (Set Yöntemi)

"Izgarayı göster" özelliği için yeni değer ayarlar.

```
bool  ShowGrid(
   bool  show      // yeni değer
   )
```

Parametreler

show

[in] "Izgarayı göster" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

ShowVolumes (Get Yöntemi)

"Hacimleri göster" özelliğinin değerini alır.

```
bool  ShowVolumes() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Hacimleri göster" özelliğinin değeri. Hiçbir çizelge atanmamışsa 'false' dönüşü yapar.

ShowVolumes (Set Yöntemi)

"Hacimleri göster" özelliği için yeni değer ayarlar.

```
bool  ShowVolumes(
   bool  show      // yeni değer
   )
```

Parametreler

show

[in] "Hacimleri göster" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

ShowObjectDescr (Get Yöntemi)

"Nesne açıklamalarını göster" özelliğinin değerini alır.

```
bool  ShowObjectDescr() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Nesne açıklamalarını göster" özelliğinin değeri. Hiçbir çizelge atanmamışsa 'false' dönüşü yapar.

ShowObjectDescr (Set Yöntemi)

"Nesne açıklamalarını göster" özelliği için yeni değer ayarlar.

```
bool  ShowObjectDescr(
   bool  show      // yeni değer
   )
```

Parametreler

show

[in] "Nesne açıklamalarını göster" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

ShowDateScale

"Zaman ölçeğini göster" özelliği için yeni değer ayarlar.

```
bool  ShowDateScale(
   bool  show      // yeni değer
   )
```

Parametreler

show

[in] "Zaman ölçeğini göster" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

ShowPriceScale

"Fiyat ölçeğini göster" özelliği için yeni değer ayarlar.

```
bool  ShowPriceScale(
   bool  show      // yeni değer
   )
```

Parametreler

show

[in] "Fiyat ölçeğini göster" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

ColorBackground (Get Yöntemi)

"Arka-plan rengi" özelliğinin değerini alır (çizelge arka-plan rengi).

```
color  ColorBackground() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Arka-plan rengi" özelliğinin değeri. Tutturulmuş bir çizelge yoksa CLR_NONE değerine dönüş yapar.

ColorBackground (Set Yöntemi)

"Arka-plan rengi" özelliğinin değerini ayarlar.

```
bool  ColorBackground(
   color  new_color      // yeni arka-plan rengi
   )
```

Parametreler

new_color

[in] Yeni arka-plan rengi.

Dönüş değeri

Başarılı ise 'true', renk değişmediyse 'false'.

ColorForeground (Get Yöntemi)

"Ön-plan rengi" özelliğinin değerini alır (Çizelgenin eksenleri, ölçeği ve OHLC dizgilerinin rengi).

```
color  ColorForeground() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Ön-plan rengi" özelliğinin değeri. Tutturulmuş bir çizelge yoksa CLR_NONE değerine dönüş yapar.

ColorForeground (Set Yöntemi)

"Ön-plan rengi" özelliği için yeni değer ayarlar (Çizelgenin eksenleri, ölçeği ve OHLC dizgilerinin rengi)

```
bool  ColorForeground(
   color  new_color      // yeni renk
   )
```

Parametreler

new_color

[in] Eksenler, ölçek ve OHLC dizgileri için yeni renk.

Dönüş değeri

Başarılı ise 'true', renk değişmediyse 'false'.

ColorGrid (Get Yöntemi)

"Izgara rengi" özelliğinin değerini alır.

```
color  ColorGrid() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Izgara rengi" özelliğinin değeri. Tutturulmuş bir çizelge yoksa CLR_NONE değerine dönüş yapar.

ColorGrid (Set Yöntemi)

"Izgara rengi" özelliği için yeni değer ayarlar.

```
bool  ColorGrid(
   color  new_color      // yeni ızgara rengi
   )
```

Parametreler

new_color

[in] Yeni ızgara rengi.

Dönüş değeri

Başarılı ise 'true', renk değişmediyse 'false'.

ColorBarUp (Get Yöntemi)

"Yükseliş çubuğu rengi" özelliğinin değerini alır (yukarı yönlü çubukların, gölgelerin ve mum gövde çizgilerinin rengi).

```
color  ColorBarUp() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Yükseliş çubuğu rengi" özelliğinin değeri. Tutturulmuş bir çizelge yoksa CLR_NONE değerine dönüş yapar.

ColorBarUp (Set Yöntemi)

"Yükseliş çubuğu rengi" özelliği için yeni değer ayarlar.

```
bool  ColorBarUp(
   color  new_color      // yükseliş çubukları için yeni renk
   )
```

Parametreler

new_color

[in] Yükseliş çubukları için yeni renk.

Dönüş değeri

Başarılı ise 'true', renk değişmediyse 'false'.

ColorBarDown (Get Yöntemi)

"Düşüş çubuğu rengi" özelliğinin değerini alır (aşağı yönlü çubukların, gölgelerin ve mum gövde çizgilerinin rengi)

```
color  ColorBarDown() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Düşüş çubuğu rengi" özelliğinin değeri. Tutturulmuş bir çizelge yoksa CLR_NONE değerine dönüş yapar.

ColorBarDown (Set Yöntemi)

"Düşüş çubuğu rengi" özelliği için yeni değer ayarlar.

```
bool  ColorBarDown(
   color  new_color      // düşüş çubukları için yeni renk
   )
```

Parametreler

new_color

[in] Düşüş çubukları için yeni renk.

Dönüş değeri

Başarılı ise 'true', renk değişmediyse 'false'.

ColorCandleBull (Get Method)

"Yükseliş mumu rengi" özelliğinin değerini alır (boğa mumlarının gövde rengi)

```
color  ColorCandleBull() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Yükseliş mumu rengi" özelliğinin değeri. Tutturulmuş bir çizelge yoksa CLR_NONE değerine dönüş yapar.

ColorCandleBull (Set Method)

"Yükseliş mumu rengi" özelliğinin değerini ayarlar.

```
bool  ColorCandleBull(
   color  new_color      // yükseliş mumu gövdesi için yeni renk
   )
```

Parametreler

new_color

[in] Yükseliş mumu gövdesinin yeni rengi.

Dönüş değeri

Başarılı ise 'true', renk değişmediyse 'false'.

ColorCandleBear (Get Method)

"Düşüş mumu rengi" özelliğinin değerini alır (ayı mumlarının gövde rengi).

```
color  ColorCandleBear() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Düşüş mumu rengi" özelliğinin değeri. Tutturulmuş bir çizelge yoksa CLR_NONE değerine dönüş yapar.

ColorCandleBear (Set Method)

"Düşüş mumu rengi" özelliğinin değerini ayarlar.

```
bool  ColorCandleBear(
   color  new_color      // düşüş mumu gövdesi için yeni renk
   )
```

Parametreler

new_color

[in] Düşüş mumu gövdesinin yeni rengi.

Dönüş değeri

Başarılı ise 'true', renk değişmediyse 'false'.

ColorChartLine (Get Yöntemi)

"Çizgi grafiği rengi" özelliğinin değerini alır (çizgi grafiği ve Doji mumlarının rengi)

```
color  ColorChartLine() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Çizgi grafiği rengi" özelliğinin değeri. Tutturulmuş bir çizelge yoksa CLR_NONE değerine dönüş yapar.

ColorChartLine (Set Yöntemi)

"Çizgi grafiği rengi" özelliği için yeni değer ayarlar.

```
bool  ColorChartLine(
   color  new_color      // çizgi grafiğinin yeni rengi
   )
```

Parametreler

new_color

[in] Çizgi grafiğinin yeni rengi.

Dönüş değeri

Başarılı ise 'true', renk değişmediyse 'false'.

ColorVolumes (Get Yöntemi)

"Hacim rengi" özelliğinin değerini alır (hacimlerin ve açık pozisyonların rengi)

```
color  ColorVolumes() const
```

Dönüş değeri

Sınıf örneğine tutturulan çizelgenin "Hacim rengi" özelliğinin değeri. Tutturulmuş bir çizelge yoksa CLR_NONE değerine dönüş yapar.

ColorVolumes (Set Yöntemi)

"Hacim rengi" özelliği için yeni değer ayarlar.

```
bool  ColorVolumes(
   color  new_color      // hacimler için yeni renk
   )
```

Parametreler

new_color

[in] Hacimlerin (açık pozisyonların seviyesi) yeni rengi.

Dönüş değeri

Başarılı ise 'true', renk değişmediyse 'false'.

ColorLineBid (Get Yöntemi)

"Satış fiyatı çizgisi rengi" özelliğinin değerini alır.

```
color  ColorLineBid() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Satış fiyatı çizgisi rengi" özelliğinin değeri. Tutturulmuş bir çizelge yoksa CLR_NONE değerine dönüş yapar.

ColorLineBid (Set Yöntemi)

"Satış fiyatı çizgisi rengi" özelliği için yeni değer ayarlar.

```
bool  ColorLineBid(
   color  new_color      // satış fiyatı çizgisi için yeni renk
   )
```

Parametreler

new_color

[in] Satış fiyatı çizgisi için yeni renk.

Dönüş değeri

Başarılı ise 'true', renk değişmediyse 'false'.

ColorLineAsk (Get Yöntemi)

"Alış fiyatı çizgisi rengi" özelliğinin değerini alır

```
color  ColorLineAsk() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Alış fiyatı çizgisi rengi" özelliğinin değeri. Tutturulmuş bir çizelge yoksa CLR_NONE değerine dönüş yapar.

ColorLineAsk (Set Yöntemi)

"Alış fiyatı çizgisi rengi" özelliği için yeni değer ayarlar.

```
bool  ColorLineAsk(
   color  new_color      // alış fiyatı çizgisi için yeni renk
   )
```

Parametreler

new_color

[in] Alış fiyatı çizgisi için yeni renk.

Dönüş değeri

Başarılı ise 'true', renk değişmediyse 'false'.

ColorLineLast (Get Yöntemi)

"Son fiyat çizgisi rengi" özelliğinin değerini alır.

```
color  ColorLineLast() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Son fiyat çizgisi rengi" özelliğinin değeri. Tutturulmuş bir çizelge yoksa CLR_NONE değerine dönüş yapar.

ColorLineLast (Set Yöntemi)

"Son fiyat çizgisi rengi" özelliği için yeni değer ayarlar.

```
bool  ColorLineLast(
   color  new_color      // son fiyat çizgisi için yeni renk
   )
```

Parametreler

new_color

[in] Son işlem fiyatı çizgisinin yeni rengi.

Dönüş değeri

Başarılı ise 'true', renk değişmediyse 'false'.

ColorStopLevels (Get Yöntemi)

"Stop seviyeleri rengi" özelliğinin değerini alır (Stop Loss ve Take Profit seviyelerinin rengi).

```
color  ColorStopLevels() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin "Stop seviyeleri rengi" özelliğinin değeri. Tutturulmuş bir çizelge yoksa CLR_NONE değerine dönüş yapar.

ColorStopLevels (Set Yöntemi)

"Stop seviyeleri rengi" özelliği için yeni değer ayarlar.

```
bool  ColorStopLevels(
   color  new_color      // Stop Loss ve Take Profit seviyelerinin yeni rengi
   )
```

Parametreler

new_color

[in] Stop Loss ve Take Profit seviyelerinin yeni rengi.

Dönüş değeri

Başarılı ise 'true', renk değişmediyse 'false'.

VisibleBars

Görünür çizelge çubuklarının toplam sayısını alır.

```
int  VisibleBars() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgedeki görünen çubukların toplam sayısı. Tutturulmuş bir çizelge yoksa, 0 dönüşü yapar.

WindowsTotal

Çizelge pencerelerinin (alt-pencereler dahil) toplam sayısını alır.

```
int  WindowsTotal() const
```

Dönüş değeri

Sınıf örneğine tutturulan çizelgedeki pencerelerin (alt-pencereler dahil) toplam sayısını. Tutturulmuş bir çizelge yoksa, 0 dönüşü yapar.

WindowIsVisible

Belirtilen alt-pencerenin görünürlük bayrağını alır.

```
bool  WindowIsVisible(
   int  num      // alt-pencere numarası
   ) const
```

Parametreler

num

[in] Alt-pencere numarası (0 değeri ana pencereyi temsil eder).

Dönüş değeri

Sınıf örneğine atanmış olan çizelge alt penceresinin görünürlük bayrağına dönüş yapar. Hiçbir çizelge atanmamışsa 'false' dönüşü yapar.

WindowHandle

Çizelgenin 'pencere tanıtıcı değerini' alır (HWND).

```
int  WindowHandle() const
```

Dönüş değeri

Sınıf örneğine tutturulmuş çizelge alt-penceresinin tanıtıcı değeri. Tutturulmuş bir çizelge yoksa INVALID_HANDLE değerine dönüş yapar.

FirstVisibleBar

Çizelgenin görünen ilk çubuğunun numarasını alır

```
int  FirstVisibleBar() const
```

Dönüş değeri

Sınıf örneğine atanan çizelgenin görünen ilk çubuğunun numarası. Tutturulmuş bir çizelge yoksa, -1 dönüşü yapar.

WidthInBars

Çubuk sayısı bazında pencere genişliğini alır.

```
int  WidthInBars() const
```

Dönüş değeri

Sınıf örneğine tutturulmuş çizelge alt-penceresinin çubuk sayısı bazında genişliği. Tutturulmuş bir çizelge yoksa, 0 dönüşü yapar.

WidthInPixels

Piksel bazında alt-pencere genişliğini alır.

```
int  WidthInPixels() const
```

Dönüş değeri

Sınıf örneğine tutturulmuş çizelge alt-penceresinin piksel bazında genişliği. Tutturulmuş bir çizelge yoksa, 0 dönüşü yapar.

HeightInPixels

Piksel bazında alt-pencere yüksekliğini alır.

```
int  HeightInPixels(
   int  num      // alt-pencere numarası
   ) const
```

Parametreler

num

[in] Alt-pencere numarası (0 değeri ana pencereyi temsil eder).

Dönüş değeri

Sınıf örneğine tutturulmuş çizelge alt-penceresinin piksel bazında yüksekliği. Tutturulmuş bir çizelge yoksa, 0 dönüşü yapar.

PriceMin

Belirtilen alt penceredeki en küçük fiyat değerini alır.

```
double  PriceMin(
   int  num      // alt-pencere numarası
   ) const
```

Parametreler

num

[in] Alt-pencere numarası (0 değeri ana pencereyi temsil eder).

Dönüş değeri

Sınıf örneğine atanan çizelgenin en küçük fiyat değeri. Atanmış bir çizelge yoksa EMPTY_VALUE dönüşü yapar.

PriceMax

Belirtilen alt penceredeki en büyük fiyat değerini alır.

```
double  PriceMax(
   int  num      // alt-pencere numarası
   ) const
```

Parametreler

num

[in] Alt-pencere numarası (0 değeri ana pencereyi temsil eder).

Dönüş değeri

Sınıf örneğine atanan çizelgenin en büyük fiyat değeri. Atanmış bir çizelge yoksa EMPTY_VALUE dönüşü yapar.

Attach

Mevcut çizelgeyi sınıf örneğine tutturur.

```
void  Attach()
```

Attach

Belirtilen çizelgeyi sınıf örneğine tutturur.

```
void  Attach(
   long  chart      // Çizelge tanımlayıcısı
   )
```

Parametreler

chart

[in] Tutturulacak çizelgenin tanımlayıcısı

FirstChart

Müşteri terminalindeki ilk çizelgeyi sınıf örneğine tutturur.

```
void  FirstChart()
```

NextChart

Müşteri terminalindeki bir sonraki çizelgeyi sınıf örneğine tutturur.

```
void  NextChart()
```

Open

Belli parametrelerle bir çizelge açar ve onu sınıf örneğine tutturur.

```
long  Open(
   const string     symbol_name,      // Sembol ismi
   ENUM_TIMEFRAMES  timeframe         // Periyot
   )
```

Parametreler

symbol_name

[in] Sembol ismi. NULL değeri mevcut çizelge sembolünü temsil eder.

timeframe

[in] Çizelge zaman-dilimi (ENUM_TIMEFRAMES sayımının değerleri). 0 değeri mevcut zaman-dilimini temsil eder.

Dönüş değeri

Çizelge tanımlayıcısı.

Detach

Çizelgeyi sınıf örneğinden ayırır.

```
void  Detach()
```

Close

Sınıf örneğine tutturulmuş çizelgeyi kapatır.

```
void  Close()
```

BringToTop

Çizelgeyi diğer çizelgelerin üzerine taşır.

```
bool  BringToTop() const
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

EventObjectCreate

Yeni nesne oluşturma olayı hakkında çizelgedeki tüm MQL5 programlarına uyarı göndermek için bir bayrak ayarlar.

```
bool  EventObjectCreate(
   bool  flag      // bayrak
   )
```

Parametreler

flag

[in] Yeni bayrak değeri.

Dönüş değeri

Başarılı ise 'true', bayrak değişmediyse 'false'.

EventObjectDelete

Nesne silme olayı hakkında çizelgedeki tüm MQL5 programlarına uyarı göndermek için bir bayrak ayarlar.

```
bool  EventObjectDelete(
   bool  flag      // bayrak
   )
```

Parametreler

flag

[in] Yeni bayrak değeri.

Dönüş değeri

Başarılı ise 'true', bayrak değişmediyse 'false'.

IndicatorAdd

Belirtilen tanıtıcı değere sahip bir göstergeyi belirtilen çizelge alt-penceresine ekler.

```
bool  IndicatorAdd(
   int   sub_win         // alt-pencere numarası
   int   handle          // gösterge tanıtıcı değeri
   );
```

Parametreler

sub_win

[in] Çizelge alt-penceresinin numarası. 0 değeri ana pencereyi temsil eder. olmayan bir alt-pencere numarası belirtilmişse yeni alt-pencere oluşturulur.

handle

[in] Gösterge tanıtıcı değeri.

Dönüş Değeri

Fonksiyon başarı durumunda 'true', aksi durumda 'false' dönüşü yapar. Hata ile ilgili daha fazla bilgi için GetLastError() çağrısı yapın.

Ayrıca Bakınız

IndicatorDelete(), IndicatorsTotal(), IndicatorName().

IndicatorDelete

Belirtilen tanıtıcı değere sahip bir göstergeyi belirtilen çizelge alt-penceresinden kaldırır.

```
bool  IndicatorDelete(
   int            sub_win       // alt-pencere numarası
   const string   name          // gösterge kısa ismi
   );
```

Parametreler

sub_win

[in] Çizelge alt-penceresinin numarası. 0 değeri ana pencereyi temsil eder.

const name

[in] IndicatorSetString() fonksiyonu kullanılarak INDICATOR_SHORTNAME özelliğinde ayarlanmış olan gösterge kısa ismi. Göstergenin kısa ismini almak için IndicatorName() fonksiyonunu kullanın.

Dönüş Değeri

Göstergenin başarıyla silinmesi durumunda 'true', aksi durumda 'false' değerini alır. Hata ile ilgili daha fazla bilgi için GetLastError() fonksiyonunu kullanın.

Not

Çizelge alt-penceresinde aynı kısa isme sahip iki gösterge olması durumunda birincisi silinir.

Çizelge içinde silinen göstergenin değerlerini kullanan başka göstergeler mevcutsa, bu göstergeler de silinir.

iCustom() ve IndicatorCreate() fonksiyonları aracılığıyla gösterge oluştururken belirtilen dosya ismi ile gösterge kısa ismini karıştırmayın. Eğer göstergenin kısa ismi açık bir yolla ayarlanmamışsa, kaynak kodunu içeren dosyanın ismi derleme sırasında belirlenecektir.

Bir göstergenin çizelgeden kaldırılması, hesaplama kısmının da terminal belleğinden silindiği anlamına gelmez. Gösterge işleyicisini tahliye etmek için IndicatorRelease() fonksiyonunu kullanın.

Gösterge kısa ismi düzgün oluşturulmalıdır. Kısa isim, INDICATOR_SHORTNAME özelliğine IndicatorSetString() fonksiyonu kullanılarak yazılacaktır. Kısa ismin, göstergenin tüm giriş giriş parametrelerinin değerlerini içermesi tavsiye edilir, çünkü IndicatorDelete() fonksiyonu ile silinecek olan göstergeler kısa isim ile tanımlanır.

Ayrıca bakınız

IndicatorAdd(), IndicatorsTotal(), IndicatorName(), iCustom(), IndicatorCreate(), IndicatorSetString().

IndicatorsTotal

Belirtilen çizelge penceresine eklenmiş tüm göstergelerin sayısına dönüş yapar.

```
int  IndicatorsTotal(
   int   sub_win     // alt-pencere numarası
   );
```

Parametreler

sub_win

[in] Çizelge alt-penceresinin numarası. 0 değeri ana pencereyi temsil eder.

Dönüş Değeri

Belirtilen çizelge penceresine eklenmiş göstergelerin sayısı. Hata ile ilgili daha fazla bilgi için GetLastError() fonksiyonunu kullanın.

Not

Bu fonksiyon, çizelgeye eklenmiş tüm göstergeler arasında arama yapılabilmesine olanak sağlar. Tüm çizelge pencerelerinin sayısı CHART_WINDOWS_TOTAL özelliğinden, GetInteger() fonksiyonu kullanılarak elde edilebilir.

Ayrıca bakınız

IndicatorAdd(), IndicatorDelete(), IndicatorsTotal(), iCustom(), IndicatorCreate(), IndicatorSetString().

IndicatorName

Belirli bir çizelge penceresi üzerinde yer alan göstergelerin listesindeki indis değerini kullanarak bir göstergenin kısa ismine dönüş yapar

```
string  IndicatorName(
   int    sub_win      // alt-pencere numarası
   int    index        // göstergenin, alt-pencereye eklenmiş göstergeler listesindeki i
   );
```

Parametreler

sub_win

[in] Çizelge alt-penceresinin numarası. 0 değeri ana pencereyi temsil eder.

index

[in] Göstergenin, göstergeler listesindeki indisi. Göstergelerin numaralandırılmasına 0 ile başlanır, yani listedeki ilk gösterge 0 indisine sahiptir. Listedeki göstergelerin toplam sayısını elde etmek için IndicatorsTotal() fonksiyonunu kullanın.

Dönüş Değeri

IndicatorSetString() fonksiyonu kullanılarak INDICATOR_SHORTNAME özelliğinde ayarlanmış olan gösterge kısa ismi. Hata detaylarını görmek için GetLastError() fonksiyonunu kullanın.

Not

iCustom() ve IndicatorCreate() fonksiyonları aracılığıyla gösterge oluştururken belirtilen dosya ismi ile gösterge kısa ismini karıştırmayın. Eğer göstergenin kısa ismi açık bir yolla ayarlanmamışsa, kaynak kodunu içeren dosyanın ismi derleme sırasında belirlenecektir.

Gösterge kısa ismi düzgün oluşturulmalıdır. Kısa isim, INDICATOR_SHORTNAME özelliğine IndicatorSetString() fonksiyonu kullanılarak yazılacaktır. Kısa ismin, göstergenin tüm giriş giriş parametrelerinin değerlerini içermesi tavsiye edilir, çünkü IndicatorDelete() fonksiyonu ile silinecek olan göstergeler kısa isim ile tanımlanır.

Ayrıca bakınız

IndicatorAdd(), IndicatorDelete, IndicatorsTotal, iCustom(), IndicatorCreate(), IndicatorSetString().

Navigate

Çizelgeyi kaydırır.

```
bool  Navigate(
   ENUM_CHART_POSITION  position,      // Konum
   int                  shift=0        // Kaydırma
   )
```

Parametreler

position

[in] ENUM_CHART_POSITION sayımının değerlerinden biri.

shift=0

[in] Kaydırma yapılacak çubuk sayısı.

Dönüş değeri

Başarılı ise 'true', çizelge kaydırılamadıysa 'false'.

Symbol

Çizelge sembolünü alır.

```
string  Symbol() const
```

Dönüş değeri

Sınıf örneğine tutturulan çizelgenin sembolü. Tutturulmuş bir çizelge yoksa, 0 dönüşü yapar.

Period

Çizelge periyodunu alır.

```
ENUM_TIMEFRAMES  Period() const
```

Dönüş değeri

Sınıf örneğine tutturulan çizelgenin periyodu. Tutturulmuş bir çizelge yoksa, 0 dönüşü yapar.

Redraw

Sınıf örneğine tutturulan çizelgeyi yeniden çizer.

```
void  Redraw()
```

GetInteger

Bu fonksiyon karşılık gelen nesne özelliğinin değerine dönüş yapar. Nesne özelliği tamsayı tipinde olmalıdır. Fonksiyonun iki versiyonu vardır.

1. Hızlı bir şekilde özellik değerine dönüş yapar.

```
long  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,          // özellik tanımlayıcısı
   int                          sub_window=0      // alt-pencere numarası
   ) const
```

2. Başarı durumunda, belirtilen tamsayı tipli özellik değerini referansla geçirilen son parametreye atar.

```
bool  GetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,        // özellik tanımlayıcısı
   int                          sub_window,     // alt-pencere numarası
   long&                        value           // özellik değerini buradan elde edece
   ) const
```

Parametreler

prop_id

[in] Özellik tanımlayıcısı (ENUM_CHART_PROPERTY_INTEGER sayımının değerlerinden biri).

sub_window

[in] Çizelge alt-pencere numarası.

value

[in] İstenen özelliğin değerini alan tamsayı tipli değişken.

Dönüş Değeri

Sınıf örneğine atanan çizelge özelliğinin değeri. Atanmış bir çizelge yoksa, -1 dönüşü yapar.

Fonksiyonun ikinci versiyonunda eğer nesne mevcutsa ve değer söz konusu referans değişkenine atanmışsa 'true', aksi durumda 'false' dönüşü yapar. Hata durumuyla ilgili daha fazla bilgi için GetLastError() çağrısını yapın.

SetInteger

Tamsayı tipli bir özellik için yeni değer ayarlar.

```
bool  SetInteger(
   ENUM_CHART_PROPERTY_INTEGER  prop_id,      // özellik tanımlayıcısı
   long                         value         // yeni değer
   )
```

Parametreler

prop_id

[in] Özellik tanımlayıcısı (ENUM_CHART_PROPERTY_INTEGER sayımının değerlerinden biri).

value

[in] Özellik için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

GetDouble

Bu fonksiyon karşılık gelen nesne özelliğinin değerine dönüş yapar. Nesne özelliği double tipinde olmalıdır. Fonksiyonun iki versiyonu vardır.

1. Hızlı bir şekilde özellik değerine dönüş yapar.

```
double  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,          // özellik tanımlayıcısı
   int                         sub_window=0     // alt-pencere numarası
   ) const
```

2. Başarı durumunda, belirtilen double tipli özellik değerini referansla geçirilen son parametreye atar.

```
bool  GetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,        // özellik tanımlayıcısı
   int                         sub_window,     // alt-pencere numarası
   double&                     value           // özellik değerini buradan elde edece
   ) const
```

Parametreler

prop_id

[in] Özellik tanımlayıcısı (ENUM_CHART_PROPERTY_DOUBLE sayımının değerlerinden biri).

sub_window

[in] Çizelge alt-pencere numarası.

value

[in] İstenen özelliğin değerini alan double tipli değişken.

Dönüş Değeri

Sınıf örneğine atanan çizelge özelliğinin değeri. Atanmış bir çizelge yoksa, EMPTY_VALUE dönüşü yapar.

Fonksiyonun ikinci versiyonunda eğer nesne mevcutsa ve değer söz konusu referans değişkenine atanmışsa 'true', aksi durumda 'false' dönüşü yapar. Hata durumuyla ilgili daha fazla bilgi için GetLastError() çağrısını yapın.

SetDouble

double tipli bir özellik için yeni değer ayarlar.

```
bool  SetDouble(
   ENUM_CHART_PROPERTY_DOUBLE  prop_id,      // özellik tanımlayıcısı
   double                      value         // yeni değer
   )
```

Parametreler

prop_id

[in] Özellik tanımlayıcısı (ENUM_CHART_PROPERTY_DOUBLE sayımının değerlerinden biri).

value

[in] Özellik için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

GetString

Bu fonksiyon karşılık gelen nesne özelliğinin değerine dönüş yapar. Nesne özelliği string tipinde olmalıdır. Fonksiyonun iki versiyonu vardır.

1. Hızlı bir şekilde özellik değerine dönüş yapar.

```
string  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id      // özellik tanımlayıcısı
   ) const
```

2. Başarı durumunda, belirtilen string tipli özellik değerini referansla geçirilen son parametreye atar.

```
bool  GetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,     // özellik tanımlayıcısı
   string&                     value        // özellik değerini buradan elde edeceğiz
   ) const
```

Parametreler

prop_id

[in] Özellik tanımlayıcısı (ENUM_CHART_PROPERTY_STRING sayımı).

sub_window

[in] Çizelge alt-pencere numarası.

value

[in] İstenen özelliğin değerini alan string tipli değişken.

Dönüş Değeri

Sınıf örneğine atanan çizelge özelliğinin değeri. Atanmış bir çizelge yoksa, "" dönüşü yapar.

Fonksiyonun ikinci versiyonunda eğer nesne mevcutsa ve değer söz konusu referans değişkenine atanmışsa 'true', aksi durumda 'false' dönüşü yapar. Hata durumuyla ilgili daha fazla bilgi için GetLastError() çağrısını yapın.

SetString

string tipli bir özellik için yeni değer ayarlar.

```
bool  SetString(
   ENUM_CHART_PROPERTY_STRING  prop_id,      // özellik tanımlayıcısı
   string                      value         // yeni özellik değeri
   )
```

Parametreler

prop_id

[in] Özellik tanımlayıcısı (ENUM_CHART_PROPERTY_STRING sayımı).

value

[in] Özellik için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

SetSymbolPeriod

Sınıf örneğine tutturulan çizelgenin sembol ve periyot değerlerini değiştirir.

```
bool  SetSymbolPeriod(
   const string      symbol_name,      // Sembol
   ENUM_TIMEFRAMES   timeframe         // Periyot
   )
```

Parametreler

symbol_name

[in] Yeni sembol ismi. NULL değeri mevcut çizelge sembolünü temsil eder.

timeframe

[in] Yeni zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri). 0 değeri mevcut zaman-dilimini temsil eder.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

ApplyTemplate

Belirtilen şablonu çizelgeye uygular.

```
bool  ApplyTemplate(
   const string  filename      // Şablon dosyasının ismi
   )
```

Parametreler

filename

[in] Şablon dosyasının ismi.

Dönüş değeri

Başarılı ise 'true', şablon uygulanamadıysa 'false'.

ScreenShot

Belirtilen çizelgenin ekran görüntüsünü alır ve bir .gif dosyasına kaydeder.

```
bool  ScreenShot(
   string           filename,                  // dosya ismi
   int              width,                     // genişlik
   int              height,                    // yükseklik
   ENUM_ALIGN_MODE  align_mode=ALIGN_RIGHT     // hizalama tipi
   ) const
```

Parametreler

filename

[in] Ekran görüntüsü için dosya ismi.

width

[in] Ekran görüntüsünün piksel bazında genişliği.

height

[in] Ekran görüntüsünün piksel bazında yüksekliği.

align_mode=ALIGN_RIGHT

[in] Hizalama tipi (ekran görüntüsü yeterince dar ise).

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

WindowOnDropped

Nesnenin (Uzman veya Script) bırakma noktasına karşılık gelen alt-pencerenin numarasını alır.

```
int  WindowOnDropped() const
```

Dönüş değeri

Nesnenin bırakma noktasına karşılık gelen alt-pencerenin numarası. 0 değeri ana çizelge penceresini temsil eder.

PriceOnDropped

Nesnenin (uzman veya script) bırakma noktasına karşılık gelen fiyat koordinatını alır.

```
double  PriceOnDropped() const
```

Dönüş değeri

Nesnenin bırakma noktasına karşılık gelen fiyat koordinatı.

TimeOnDropped

Nesnenin (Uzman veya Script) bırakma noktasına karşılık gelen zaman koordinatını alır.

```
datetime  TimeOnDropped() const
```

Dönüş değeri

Nesnenin bırakma noktasının zaman koordinatı.

XOnDropped

Nesnenin (Uzman veya Script) bırakma noktasına karşılık gelen X koordinatını alır.

```
int  XOnDropped() const
```

Dönüş değeri

Nesnenin bırakma noktasının X koordinatı.

YOnDropped

Nesnenin (Uzman veya Script) bırakma noktasına karşılık gelen Y koordinatını alır.

```
int  YOnDropped() const
```

Dönüş değeri

Nesnenin bırakma noktasının Y koordinatı.

Save

Nesne parametrelerini dosyaya kaydeder.

```
virtual bool  Save(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Dosyadan nesne parametreleri yükler.

```
virtual bool  Load(
   int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] FileOpen(...) fonksiyonu ile açılmış olan dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Type

Grafik nesnesinin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CChart için 0x1111).

Bilimsel Çizelgeler

Grafikler kitaplığı özel grafiklerin hızlı çizimi için gereken global fonksiyonları içerir. Eksenlerin ve eğrilerin çizilmesini sağlayan yöntemlerin yanında grafik özelliklerinin değiştirilmesi için gereken hızlı erişim çözümleri de bu kitaplık içerisinde sunulmuştur.

Kütüphaneyle çalışmaya başlamak için Görselleştirin! R dilinin 'plot' fonksiyonuna benzer MQL5 grafik kütüphanesi.

Grafikler kitaplığı terminal çalışma dizininde, Include\Graphics klasöründe yer alır.

Sınıf	Açıklama
CAxis	Koordinat eksenleriyle çalışmak için kullanılan sınıf
ColorGenerator	Varsayılan renk şemasını tanımlayan sınıf
CCurve	Eğrilerle çalışmak için kullanılan sınıf
CGraphic	Özel çizelgeler oluşturmak için temel sınıf

GraphPlot

Hızlı eğri çizimi için fonksiyonlar.

Y koordinatlarıyla eğri çizimi için kullanılan versiyon.

```
string  GraphPlot(
   const double      &y[],                  // Y koordinatları
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // eğri tipi
   )
```

Not

Y dizisinin indisleri eğri çizimi sırasında X koordinatları olarak kullanılır.

Y ve X koordinatlarıyla eğri çizimi için kullanılan versiyon.

```
string  GraphPlot(
   const double      &x[],                  // X koordinatları
   const double      &y[],                  // Y koordinatları
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // eğri tipi
   )
```

Y ve X koordinatlarıyla iki eğri çizimi için kullanılan versiyon.

```
string  GraphPlot(
   const double      &x1[],                 // X koordinatları
   const double      &y1[],                 // Y koordinatları
   const double      &x2[],                 // X koordinatları
   const double      &y2[],                 // Y koordinatları
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // eğri tipi
   )
```

Y ve X koordinatlarıyla üç eğri çizimi için kullanılan versiyon.

```
string  GraphPlot(
   const double      &x1[],                 // X koordinatları
   const double      &y1[],                 // Y koordinatları
   const double      &x2[],                 // X koordinatları
   const double      &y2[],                 // Y koordinatları
   const double      &x3[],                 // X koordinatları
   const double      &y3[],                 // Y koordinatları
   ENUM_CURVE_TYPE   type=CURVE_POINTS      // eğri tipi
   )
```

CPoint2D noktalarının koordinatlarıyla eğri çizimi için kullanılan versiyon.

```
string  GraphPlot(
   const CPoint2D   &points[],            // eğri koordinatları
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // eğri tipi
   )
```

CPoint2D noktalarının koordinatlarıyla iki eğrinin çizimi için kullanılan versiyon.

```
string  GraphPlot(
   const CPoint2D   &points1[],           // eğri koordinatları
   const CPoint2D   &points2[],           // eğri koordinatları
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // eğri tipi
   )
```

CPoint2D noktalarının koordinatlarıyla üç eğrinin çizimi için kullanılan versiyon.

```
string  GraphPlot(
   const CPoint2D   &points1[],           // eğri koordinatları
   const CPoint2D   &points2[],           // eğri koordinatları
   const CPoint2D   &points3[],           // eğri koordinatları
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // eğri tipi
   )
```

CurveFunction işaretçisi ile eğri çizimi yapılan versiyon.

```
string  GraphPlot(
   CurveFunction    function,             // fonksiyon işaretçisi
   const double     from,                 // argümanın başlangıç değeri
   const double     to,                   // argümanın son değeri
   const double     step,                 // argümanın artış değeri
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // eğri tipi
   )
```

CurveFunction işaretçisi ile iki eğri çizimi yapılan versiyon.

```
string  GraphPlot(
   CurveFunction    function1,            // fonksiyon işaretçisi
   CurveFunction    function2,            // fonksiyon işaretçisi
   const double     from,                 // argümanın başlangıç değeri
   const double     to,                   // argümanın son değeri
   const double     step,                 // argümanın artış değeri
   ENUM_CURVE_TYPE  type=CURVE_POINTS     // eğri tipi
   )
```

CurveFunction işaretçisi ile üç eğri çizimi yapılan versiyon.

```
string  GraphPlot(
   CurveFunction    function1,             // fonksiyon işaretçisi
   CurveFunction    function2,             // fonksiyon işaretçisi
   CurveFunction    function3,             // fonksiyon işaretçisi
   const double     from,                  // argümanın başlangıç değeri
   const double     to,                    // argümanın son değeri
   const double     step,                  // argümanın artış değeri
   ENUM_CURVE_TYPE  type=CURVE_POINTS      // eğri tipi
   )
```

Parametreler

&x[]

[in] X koordinatları.

&y[]

[in] Y koordinatları.

&x1[]

[in] İlk eğri için X koordinatları.

&y1[]

[in] İlk eğri için Y koordinatları.

&x2[]

[in] İkinci eğri için X koordinatları.

&y2[]

[in] İkinci eğri için Y koordinatları.

&x3[]

[in] Üçüncü eğri için X koordinatları.

&y3[]

[in] Üçüncü eğri için Y koordinatları.

&points[]

[in] Eğri noktalarının koordinatlari.

&points1[]

[in] İlk eğrinin noktalarının koordinatlari.

&points2[]

[in] İkinci eğrinin noktalarının koordinatlari.

&points3[]

[in] Üçüncü eğrinin noktalarının koordinatlari.

function

[in] CurveFunction fonksiyonunun işaretçisi.

function1

[in] İlk fonksiyonun işaretçisi.

function2

[in] İkinci fonksiyonun işaretçisi.

function3

[in] Üçüncü fonksiyonun işaretçisi.

from

[in] İlk X koordinatına karşılık gelir.

to

[in] Son X koordinatına karşılık gelir.

step

[in] X koordinatlarının hesaplanması için gereken parametre.

type=CURVE_POINTS

[in] Eğri tipi.

Dönüş Değeri

Grafiksel kaynağın ismi.

CAxis

CAxis koordinat eksenleriyle çalışmak için tasarlanmış grafik kütüphanesidir.

Açıklama

CAxis sınıfı koordinat eksenleriyle ilgili çeşitli parametreleri depolar. Koordinat eksenlerinin dinamik olarak ölçeklenmesini sağlar.

Bildirim

```
class CAxis
```

Başlık

```
#include <Graphics\Axis.mqh>
```

Sınıf yöntemleri

Yöntem	Açıklama
AutoScale	Otomatik ölçekleme bayrağını alır/ayarlar
Min	İzin verilen en düşük eksen değerini alır/ayarlar
Max	İzin verilen en yüksek eksen değerini alır/ayarlar
Step	Eksenin adım değerini alır
Name	Eksen ismini alır/ayarlar
Color	Eksen rengini alır/ayarlar
ValuesSize	Eksen sayılarının boyutunu alır/ayarlar
ValuesWidth	Eksen sayılarının maksimum görünürlük değerini alır/ayarlar
ValuesFormat	Eksen sayılarının biçimini alır/ayarlar
ValuesDateTimeMode	Tarihi metne dönüştürme biçimini alır.
ValuesFunctionFormat	Eksen değerlerinin gösterilme biçimini tanımlayan fonksiyonun işaretçisini alır.
ValuesFunctionFormatCBData	Eksen değerlerinin dönüştürülmesiyle ilgili ek veriler içerebilen nesnenin işaretçisini alır.
NameSize	Eksen isminin yazı-tipi boyutunu alır/ayarlar
ZeroLever	"Sıfır dahili" değerini alır/ayarlar
DefaultStep	Eksenin başlangıç adım değerini alır/ayarlar
MaxLabels	Eksen için izin verilen en fazla sayı miktarını alır/ayarlar

Yöntem	Açıklama
MinGrace	Eksen minimumu için "tolerans" değerini alır/ayarlar
MaxGrace	Eksen maksimumu için "tolerans" değerini alır/ayarlar
SelectAxisScale	Eseni otomatik boyutlandırır.

AutoScale (Get yöntemi)

Otomatik ölçeklendirme gereksinimi için belirtilen bayrağa dönüş yapar.

```
bool  AutoScale()
```

Dönüş Değeri

Bayrak değeri.

Not

true — otomatik ölçeklendirme yap.

false — otomatik ölçeklendirme yapma.

AutoScale (Set Yöntemi)

Otomatik ölçeklendirme gereksinimi için belirtilen bayrağı ayarlar.

```
void  AutoScale(
   const bool  auto      // bayrak değeri
   )
```

Parametreler

auto

[in]

Not

true — otomatik ölçeklendirme yap.

false — otomatik ölçeklendirme yapma.

Min (Get yöntemi)

Minimum eksen değerine dönüş yapar.

```
double  Min()
```

Dönüş Değeri

Minimum eksen değeri

Min (Set yöntemi)

Minimum eksen değerini ayarlar.

```
void  Min(
   const double  min      // minimum değer
   )
```

Parametreler

min

[in] Minimum değer.

Max (Get yöntemi)

Maksimum eksen değerine dönüş yapar.

```
double  Max()
```

Dönüş Değeri

Maksimum eksen değeri.

Max (Set yöntemi)

Maksimum eksen değerini ayarlar

```
void  Max(
   const double  max       // maksimum değer
   )
```

Parametreler

max

[in] Maksimum eksen değeri.

Step (Get yöntemi)

Eksendeki adım değerine dönüş yapar.

```
double  Step()
```

Dönüş Değeri

Adım değeri.

Name (Get yöntemi)

Eksen ismine dönüş yapar.

```
string  Name()
```

Dönüş Değeri

Eksen ismi.

Name (Set yöntemi)

Eksen ismini ayarlar.

```
void  Name(
   const string  name      // eksen ismi
   )
```

Parametreler

name

[in] Eksen ismi.

Color (Get yöntemi)

Eksen rengine dönüş yapar.

```
color  Color()
```

Dönüş Değeri

Eksen rengi.

Color (Set yöntemi)

Eksen rengini ayarlar.

```
void  Color(
   const color  clr      // eksen color
   )
```

Parametreler

clr

[in] Eksen rengi.

ValuesSize (Get yöntemi)

Eksen sayılarının boyutuna dönüş yapar.

```
int  ValuesSize()
```

Dönüş Değeri

Eksen sayılarının boyutu.

ValuesSize (Set yöntemi)

Eksen sayılarının boyutunu ayarlar.

```
void  ValuesSize(
   const int  size      // eksen sayılarının boyutu
   )
```

Parametreler

size

[in] Eksen sayılarının boyutu

ValuesWidth (Get yöntemi)

Eksen sayılarının görüntülenmesi için izin verilen maksimum genişliğe piksel cinsinden dönüş yapar.

```
int  ValuesWidth()
```

Dönüş Değeri

Piksel cinsinden eksen sayılarının boyutu.

Not

Belirtilen sayının genişliği izin verilen maksimum genişliği aşarsa, sayı kısa kesilir ve sonuna noktalar eklenir.

ValuesWidth (Set yöntemi)

Eksen sayılarının görüntülenmesi için izin verilen maksimum genişliği piksel cinsinden ayarlar.

```
void  ValuesWidth(
   const int  width      // piksel cinsinden izin verilen maksimum genişlik
   )
```

Parametreler

width

[in] Eksen sayıları için izin verilen maksimum genişlik.

Not

Belirtilen sayının genişliği izin verilen maksimum genişliği aşarsa, sayı kısa kesilir ve sonuna noktalar eklenir.

ValuesFormat (Get yöntemi)

Eksen sayılarının biçimine dönüş yapar.

```
string  ValuesFormat()
```

Dönüş Değeri

Sayı biçimi.

ValuesFormat (Set yöntemi)

Eksen sayılarının biçimini ayarlar

```
void  ValuesFormat(
   const string  format      // eksen sayılarının biçimi
   )
```

Parametreler

format

[in] Eksen sayılarının biçimi.

ValuesDateTimeMode (Get yöntemi)

Tarihi metne dönüştürme biçimini alır.

```
int  ValuesDateTimeMode()
```

Dönüş Değeri

Tarihi metne dönüştürme biçimini.

ValuesDateTimeMode (Set yöntemi)

Tarihi metne dönüştürme biçimini ayarlar.

```
void  ValuesDateTimeMode(
   const int  mode      // tarihi metne dönüştürme biçimi
   )
```

Parametreler

mode

[in] Dönüştürme biçimi.

Not

Tarihi metne dönüştürme biçimleriyle ilgili daha fazlasını TimeToString() fonksiyonunun açıklamasında bulabilirsiniz.

ValuesFunctionFormat (Get yöntemi)

Eksen değerlerinin gösterilme biçimini tanımlayan fonksiyonun işaretçisini alır.

```
DoubleToStringFunction  ValuesFunctionFormat()
```

Dönüş Değeri

Eksen değerlerinin gösterilme biçimini tanımlayan fonksiyonun işaretçisi.

ValuesFunctionFormat (Set yöntemi)

Eksen değerlerinin gösterilme biçimini tanımlayan fonksiyonun işaretçisini ayarlar.

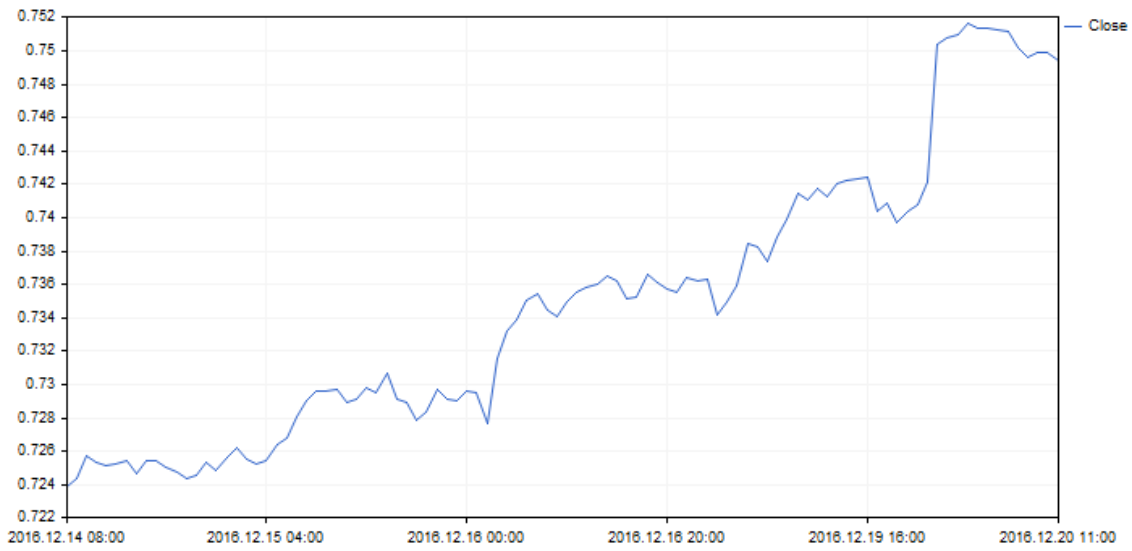
```
void  ValuesFunctionFormat(
   DoubleToStringFunction  func      // nümerik değerleri dizgi biçimine dönüştüren fo
   )
```

Parametreler

func

[in] Nümerik değerleri dizgi biçimine dönüştüren özel fonksiyon.

Örnek:



X ekseni değerlerinin gösterilme şekli şu kodla değiştirilebilir:

```
//+------------------------------------------------------------------+
//|                                                DateAxisGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//--- verileri depolamak için bir dizi
double arrX[];
double arrY[];
//+------------------------------------------------------------------+
//| X-ekseni üzerinde değerler oluşturan özel fonksiyon              |
//+------------------------------------------------------------------+
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;
     }
//--- grafiği oluştur
   CGraphic graphic;
   if(!graphic.Create(0,"DateAxisGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"DateAxisGraphic");
     }
//--- eğriyi oluştur
   CCurve *curve=graphic.CurveAdd(arrY,CURVE_LINES);
//--- X-eksenini al
   CAxis *xAxis=graphic.XAxis();
//--- X-ekseninin özelliklerini ayarla
   xAxis.AutoScale(false);
   xAxis.Type(AXIS_TYPE_CUSTOM);
   xAxis.ValuesFunctionFormat(TimeFormat);
   xAxis.DefaultStep(20.0);
//--- çiz
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

ValuesFunctionFormatCBData (Get yöntemi)

Eksen değerlerinin dönüştürülmesiyle ilgili ek veriler içerebilen nesnenin işaretçisini alır.

```
void*  ValuesFunctionFormatCBData()
```

Dönüş Değeri

Eksen değerlerinin dönüştürülmesiyle ilgili ek veriler içerebilen nesne işaretçisini alır.

ValuesFunctionFormatCBData (Set yöntemi)

Eksen değerlerinin dönüştürülmesiyle ilgili ek veriler içerebilen nesnenin işaretçisini ayarlar.

```
void  ValuesFunctionFormatCBData(
   void*  cbdata      // sınıf nesnesinin işaretçisi
   )
```

Parametreler

cbdata

[in] Eksen değerlerinin dönüştürülmesiyle ilgili ek veriler içerebilen nesnenin işaretçisi

NameSize (Get yöntemi)

Eksen isminin yazı tipi boyutuna dönüş yapar.

```
int  NameSize()
```

Dönüş Değeri

Eksen isminin yazı tipi boyutu

NameSize (Set yöntemi)

Eksen isminin yazı tipi boyutunu ayarlar

```
void  NameSize(
   const int  size      // eksen isminin yazı tipi boyutu
   )
```

Parametreler

size

[in] Eksen isminin yazı tipi boyutu.

ZeroLever (Get yöntemi)

"zero lever" (sıfır dahili) değerine dönüş yapar

```
double  ZeroLever()
```

Dönüş Değeri

"Zero lever".

Not

Bu değer sıfır sayısının eksenin ölçek aralığına nezaman dahil edileceğini belirtir.

ZeroLever (Set yöntemi)

"zero lever" (sıfır dahili) değerini ayarlar.

```
void  ZeroLever(
   const double  value      // "zero lever" değeri
   )
```

Parametreler

value

[in] "Zero lever" değeri.

Not

Bu değer sıfır sayısının eksenin ölçek aralığına nezaman dahil edileceğini belirtir.

DefaultStep (Get yöntemi)

Eksen üzerindeki başlangıç adım değerine dönüş yapar

```
double  DefaultStep()
```

Dönüş Değeri

Eksen üzerindeki adım değeri.

DefaultStep (Set yöntemi)

Eksen üzerindeki başlangıç adım değerini ayarlar

```
void  DefaultStep(
   const double  value      // eksen adım değeri
   )
```

Parametreler

value

[in] Eksendeki başlangıç adım değeri.

MaxLabels (Get yöntemi)

Eksen üzerindeki izin verilen maksimum görünür sayı miktarına dönüş yapar.

```
double  MaxLabels()
```

Dönüş Değeri

Eksen üzerindeki maksimum görünür sayı miktarı.

MaxLabels (Set yöntemi)

Eksen üzerindeki izin verilen maksimum görünür sayı miktarını ayarlar.

```
void  MaxLabels(
   const double  value      // maksimum sayı
   )
```

Parametreler

value

[in] Eksen üzerindeki izin verilen maksimum görünür sayı miktarı

MinGrace (Get yöntemi)

Eksen minimumu için kullanılan "tolerans" değerine dönüş yapar.

```
double  MinGrace()
```

Dönüş Değeri

Eksen minimumu için "tolerans" değeri

Not

Bu değer tüm eksen uzunluğunun bir bölümüyle ifade edilir. Örneğin, eksen değerleri 4.0 ile 16.0 arasında değişiyorsa eksen uzunluğu 12.0 olur. MinGrace değeri 0.1 ise, eksen uzunluğunun 10%'u (1.2) minimum değerden çıkarılır. Sonuç olarak eksen 2.8'den 16.0'a kadar olan sayıları kapsar.

MinGrace (Set yöntemi)

Eksen minimumu için kullanılacak "tolerans" değerini ayarlar.

```
void  MinGrace(
   const double  value      // "tolerans" değeri
   )
```

Parametreler

value

[in] Eksen minimumu için kullanılan tolerans değeri.

Not

Bu değer tüm eksen uzunluğunun bir bölümüyle ifade edilir. Örneğin, eksen değerleri 4.0 ile 16.0 arasında değişiyorsa eksen uzunluğu 12.0 olur. MinGrace değeri 0.1 ise, eksen uzunluğunun 10%'u (1.2) minimum değerden çıkarılır. Sonuç olarak eksen 2.8'den 16.0'a kadar olan sayıları kapsar.

MaxGrace (Get yöntemi)

Eksen maksimumu için kullanılan "tolerans" değerine dönüş yapar.

```
double  MaxGrace()
```

Dönüş Değeri

Eksen maksimumu için "tolerans" değeri.

Not

Bu değer tüm eksen uzunluğunun bir bölümüyle ifade edilir. Örneğin, eksen değerleri 4.0 ile 16.0 arasında değişiyorsa eksen uzunluğu 12.0 olur. MaxGrace değeri 0.1 ise, eksen uzunluğunun 10%'u (1.2) maksimum değere eklenir. Sonuç olarak eksen 4.0'den 17.2'ye kadar olan sayıları kapsar.

MaxGrace (Set yöntemi)

Eksen maksimumu için kullanılacak "tolerans" değerini ayarlar.

```
void  MaxGrace(
   const double  value      // "tolerans" değeri
   )
```

Parametreler

value

[in] Eksen maksimumu için kullanılacak "tolerans" değeri.

Not

Bu değer tüm eksen uzunluğunun bir bölümüyle ifade edilir. Örneğin, eksen değerleri 4.0 ile 16.0 arasında değişiyorsa eksen uzunluğu 12.0 olur. MinGrace değeri 0.1 ise, eksen uzunluğunun 10%'u (1.2) minimum değerden çıkarılır. Sonuç olarak eksen 2.8'den 16.0'a kadar olan sayıları kapsar.

SelectAxisScale

Eseni otomatik boyutlandırır.

```
void  SelectAxisScale()
```

CColorGenerator

CColorGenerator, renk paletiyle çalışmak için tasarlanmış bir yardımcı sınıftır.

Açıklama

CColorGenerator sınıfı eğriler için varsayılan başlangıç renk paletini içerir (kullanıcı ayrı bir renk belirtmemişse).

Paletteki tüm renklerin kullanılması durumunda otomatik olarak yeni renkler oluşturulur ve palet bu yeni renklerle doldurulur.

Bildirim

```
class CColorGenerator
```

Başlık

```
#include <Graphics\ColorGenerator.mqh>
```

Sınıf yöntemleri

Yöntem	Açıklama
Next	Paletteki bir sonraki renge dönüş yapar
Reset	Oluşturucuyu sıfırlar

Next

Palet üzerindeki bir sonraki renge dönüş yapar.

```
uint  Next()
```

Dönüş Değeri

Renk.

Not

Paletteki tüm renklerin kullanılması durumunda otomatik olarak yeni renkler oluşturulur ve palet bu yeni renklerle doldurulur.

Reset

Oluşturucuyu sıfırlar.

```
void  Reset()
```

CCurve

CCurve sınıfı çizelge üzerinde oluşturulan eğrilerin özellikleriyle çalışmak için tasarlanmıştır.

Açıklama

CCurve sınıfı, CGraphic sınıfıyla çalışırken gerekli koordinatları ve grafik özelliklerini alır, yükler ve ayarlar.

Üç tür eğri çizim modu bulunur: noktalar, çizgiler ve histogram. Sınıf içinde her bir çizim modu için farklı parametreler kullanılır.

Bildirim

```
class CCurve : public CObject
```

Başlık

```
#include <Graphics\Curve.mqh>
```

Kalıtım hiyerarşisi

CObject

CCurve

Sınıf yöntemleri

Yöntem	Açıklama
Type	Eğri tipine dönüş yapar
Name	Eğri ismine dönüş yapar
Color	Eğri rengine dönüş yapar
XMax	X fonksiyonunun maksimum değerini alır
XMin	X fonksiyonunun minimum değerini alır
YMax	Y fonksiyonunun maksimum değerini alır
YMin	Y fonksiyonunun minimum değerini alır
Size	Eğriyi oluşturan noktaların sayısını alır
PointsSize	Eğriyi oluşturan noktaların lineer boyutunu alır/ayarlar
PointsFill	Eğriyi oluşturan noktaların dolgu bayrağını alır/ayarlar
PointsColor	Nokta dolgu rengini alır/ayarlar
GetX	Eğrideki tüm noktaların X koordinatlarını diziye yazar
GetY	Eğrideki tüm noktaların Y koordinatlarını diziye yazar

Yöntem	Açıklama
LinesStyle	Çizgi grafikler için çizgi stilini alır/ayarlar
LinesIsSmooth	Çizgi grafikler için düzleştirme bayrağını alır/ayarlar
LinesSmoothTension	Çizgi grafikler için eğri düzleştirme parametresini alır/ayarlar
LinesSmoothStep	Çizgi grafikler oluştururken düzleştirme için kullanılacak yaklaşık çizgilerin uzunluğunu alır/ayarlar.
LinesWidth	Çizgilerle çizilen eğriler için çizgi kalınlığını alır/ayarlar
HistogramWidth	Histogramlar için sütun genişliği değerini alır/ayarlar
CustomPlotCBData	Kullanıcı tanımlı eğri çizimi için kullanılan nesnenin işaretçisini alır/ayarlar.
CustomPlotFunction	Kullanıcı tanımlı eğri çizimi için kullanılan fonksiyonun işaretçisini alır/ayarlar.
PointsType	Noktalı eğri çizerken kullanılan noktaların biçim bayrağını alır/ayarlar.
StepsDimension	Adım-tipli eğri düzleştirmede kullanılan boyut değerini alır/ayarlar.
TrendLineCoefficients	Diziye yazmak için trend çizgisi oranlarını alır/ayarlar.
TrendLineColor	Eğri için kullanılacak trend çizgisinin rengini alır/ayarlar.
TrendLineVisible	Trend çizgisi görünürlük bayrağını alır/ayarlar.
Update	Eğri koordinatlarını ayarlar.
Visible	Fonksiyonun çizelge üzerindeki görünürlüğünü ayarlayan bayrağı alır/ayarlar.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Compare

Tip

Eğri tipine dönüş yapar.

```
ENUM_CURVE_TYPE   Type()
```

Dönüş Değeri

Eğri tipi.

Name

Eğri ismine dönüş yapar.

```
string  Name()
```

Dönüş Değeri

Eğri ismi.

Color

Eğri rengine dönüş yapar.

```
uint  Color()
```

Dönüş Değeri

Eğri rengi.

XMax

X fonksiyonunun maksimum değerine (reel sayı cinsinden) dönüş yapar.

```
double   XMax()
```

Dönüş Değeri

Fonksiyonunun argümanları arasında maksimum değere sahip olan reel sayı.

XMin

X fonksiyonunun minimum değerine (reel sayı cinsinden) dönüş yapar.

```
double  XMin()
```

Dönüş Değeri

Fonksiyonunun argümanları arasında minimum değere sahip olan reel sayı.

YMax

Y fonksiyonunun maksimum değerine (reel sayı cinsinden) dönüş yapar.

```
double   YMax()
```

Dönüş Değeri

Y fonksiyonunun maksimum reel değeri.

YMin

Y fonksiyonunun minimum değerine (reel sayı cinsinden) dönüş yapar.

```
double  YMin()
```

Dönüş Değeri

Y fonksiyonunun minimum reel değeri.

Size

Eğriyi oluşturan noktaların sayısına dönüş yapar.

```
int  Size()
```

Dönüş Değeri

Eğriyi oluşturan noktaların sayısı.

PointsSize (Get yöntemi)

Eğriyi oluşturan noktaların lineer boyutuna piksel cinsinden dönüş yapar.

```
int  PointsSize()
```

Dönüş Değeri

Eğriyi oluşturan noktaların piksel cinsinden lineer boyutu.

PointsSize (Set yöntemi)

Eğriyi oluşturan noktaların lineer boyutunu piksel cinsinden ayarlar.

```
void  PointsSize(
   const int  size      // piksel cinsinden nokta boyutu
   )
```

Parametreler

size

[in] Eğriyi oluşturan noktaların piksel cinsinden lineer boyutu.

PointsFill (Get yöntemi)

Eğriyi oluşturan noktalar için dolgu işlemi gerekliliğini gösteren bayrağa dönüş yapar.

```
bool  PointsFill ()
```

Dönüş Değeri

Bayrak değeri.

Not

true — dolgu yap

false — dolgu yapma

PointsFill (Set yöntemi)

Eğriyi oluşturan noktalar için dolgu işlemi gerekliliğini gösteren bayrağı ayarlar.

```
void  PointsFill(
   const bool  fill      // bayrak değeri
   )
```

Parametreler

fill

[in] Bayrak değeri.

Not

true — dolgu yap

false — dolgu yapma

PointsColor (Get yöntemi)

Nokta dolgu rengine dönüş yapar.

```
uint  PointsColor ()
```

Dönüş Değeri

Eğriyi oluşturan noktaların dolgu rengi

PointsColor (Set yöntemi)

Nokta dolgu rengini ayarlar

```
void  PointsColor(
   const uint  clr      //nokta doldurma rengi
   )
```

Parametreler

clr

[in] Eğriyi oluşturan noktaların dolgu rengi.

GetX

Eğrideki tüm noktaların X koordinatlarını diziye yazar.

```
void  GetX(
   double  &x[]      // X koordinatları
   )
```

Parametreler

&x[]

[out] X koordinatlarının yazılacağı dizi.

GetY

Eğrideki tüm noktaların Y koordinatlarını bir diziye yazar.

```
void  GetY(
   double  &y[]      // Y koordinatları
   )
```

Parametreler

&y[]

[out] Y koordinatlarının yazılacağı dizi.

LinesStyle (Get yöntemi)

Çizgi grafikler için kullanılacak çizgi stiline dönüş yapar.

```
ENUM_LINE_STYLE  LinesStyle()
```

Dönüş Değeri

Çizgi stili.

LinesStyle (Set yöntemi)

Çizgi grafikler için kullanılacak çizgi stilini ayarlar

```
void  LinesStyle (
   ENUM_LINE_STYLE  style      // çizgi stili
   )
```

Parametreler

style

[in] Çizgi stili.

LinesIsSmooth (Get yöntemi)

Çizgi grafikler için düzleştirme gerekliliğini tanımlayan bayrağa dönüş yapar.

```
bool  LinesIsSmooth()
```

Dönüş Değeri

Bayrak değeri

Not

true — düzleştir

false — düzleştirme

LinesIsSmooth (Set yöntemi)

Çizgi grafikler için düzleştirme gerekliliğini tanımlayan bayrağı ayarlar.

```
void  LinesIsSmooth(
   const bool  smooth      // bayrak değeri
   )
```

Parametreler

smooth

[in] Bayrak değeri

Not

true — düzleştir

false — düzleştirme

LinesSmoothTension (Get yöntemi)

Çizgi grafikler için eğri düzleştirme parametresine dönüş yapar.

```
double  LinesSmoothTension()
```

Dönüş Değeri

Düzleştirme parametresinin değeri

Not

'tension' değeri (0.0; 1.0] aralığında yer alır.

LinesSmoothTension (Set yöntemi)

Çizgi grafikler için eğri düzleştirme parametresini ayarlar.

```
void  LinesSmoothTension(
   const double  tension      // parametre değeri
   )
```

Parametreler

tension

[in] Düzleştirme parametresinin değeri.

Not

'tension' değeri (0.0; 1.0] aralığında yer alır.

LinesSmoothStep (Get yöntemi)

Çizgi grafikler oluştururken düzleştirme için kullanılacak yaklaşık çizgilerin uzunluğuna dönüş yapar.

```
double  LinesSmoothStep()
```

Dönüş Değeri

Piksel cinsinden yakınsama çizgilerinin uzunluğu.

LinesSmoothStep (Set yöntemi)

Çizgi grafikler oluştururken düzleştirme için kullanılacak yaklaşık çizgilerin uzunluğunu ayarlar.

```
void  LinesSmoothStep(
   const double  step      // çizgi uzunluğu
   )
```

Parametreler

step

[in] Yakınsama çizgilerinin uzunluğu

LinesWidth (Get yöntemi)

Çizgilerle çizilen eğriler için çizgi kalınlığını alır.

```
int  LinesWidth()
```

Dönüş Değeri

Çizgi kalınlığı.

LinesWidth (Set yöntemi)

Çizgilerle çizilen eğriler için çizgi kalınlığını ayarlar

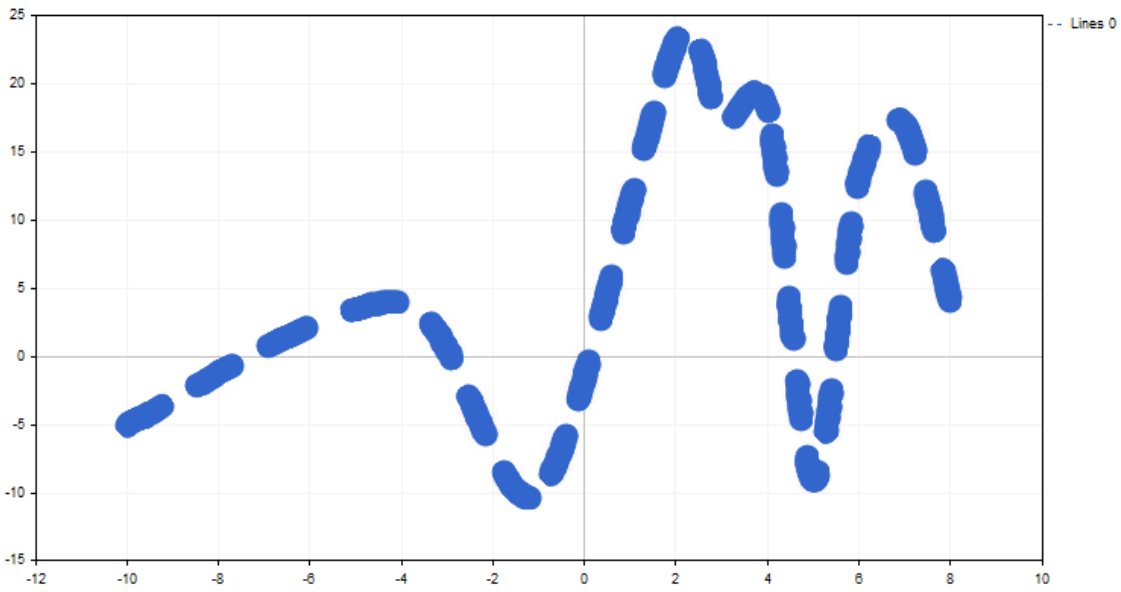
```
void  LinesWidth(
   const int  width      // çizgi kalınlığı
   )
```

Parametreler

width

[in] Çizgilerle çizilen eğriler için çizgi kalınlığı.

Örnek:



Çizgi kalınlığı aşağıdaki kod ile değiştirilebilir:

```
//+------------------------------------------------------------------+
//|                                                CandleGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
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 };
//--- grafiği oluştur
   CGraphic graphic;
   if(!graphic.Create(0,"ThickLineGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"ThickLineGraphic");
     }
//--- eğriyi oluştur
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_LINES);
//--- eğri özelliklerini ayarla
   curve.LinesSmooth(true);
   curve.LinesStyle(STYLE_DASH);
   curve.LinesEndStyle(LINE_END_ROUND);
   curve.LinesWidth(10);
//--- çiz
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

LinesEndStyle (Set yöntemi)

Eğri çizimi için çizgiler kullanırken çizgi sonu stilini belirten bayrağın değerini alır.

```
ENUM_LINE_END  LinesEndStyle()
```

Dönüş Değeri

Eğri çizimi için çizgiler kullanırken çizgi sonu stilini belirten bayrağın değeri.

LinesEndStyle (Get yöntemi)

Eğri çizimi için çizgiler kullanırken çizgi sonu stilini belirten bayrağın değerini ayarlar.

```
void  LinesEndStyle(
   ENUM_LINE_END  end_style      // bayrak değeri
   )
```

Parametreler

end_style

[in] Eğri çizimi için çizgiler kullanırken çizgi sonu stilini belirten bayrağın değeri.

HistogramWidth (Get yöntemi)

Histogramlar için sütun genişliği değerine dönüş yapar

```
int  HistogramWidth()
```

Dönüş Değeri

Piksel cinsinden sütun genişliği.

HistogramWidth (Set yöntemi)

Histogramlar için sütun genişliği değerini ayarlar.

```
void  HistogramWidth(
   const int  width      // sütun genişliği
   )
```

Parametreler

width

[in] Piksel cinsinden sütun genişliği.

CustomPlotCBData (Get yöntemi)

Kullanıcı tanımlı eğri çizimi için kullanılan nesnenin işaretçisini alır.

```
void*  CustomPlotCBData()
```

Dönüş Değeri

Kullanıcı tanımlı eğri çizimi için kullanılan nesnenin işaretçisi.

CustomPlotCBData (Set yöntemi)

Kullanıcı tanımlı eğri çizimi için kullanılan nesnenin işaretçisini ayarlar.

```
void  CustomPlotCBData(
   void*  cbdata      // nesne işaretçisi
   )
```

Parametreler

cbdata

[in] Kullanıcı tanımlı eğri çizimi için kullanılan nesnenin işaretçisi

CustomPlotFunction (Get yöntemi)

Kullanıcı tanımlı eğri çizimi için kullanılan fonksiyonun işaretçisini alır.

```
PlotFucntion  CustomPlotFunction()
```

Dönüş Değeri

Kullanıcı tanımlı eğri çizimi için kullanılan fonksiyonun işaretçisi.

CustomPlotFunction (Set yöntemi)

Kullanıcı tanımlı eğri çizimi için kullanılan fonksiyonun işaretçisini ayarlar.

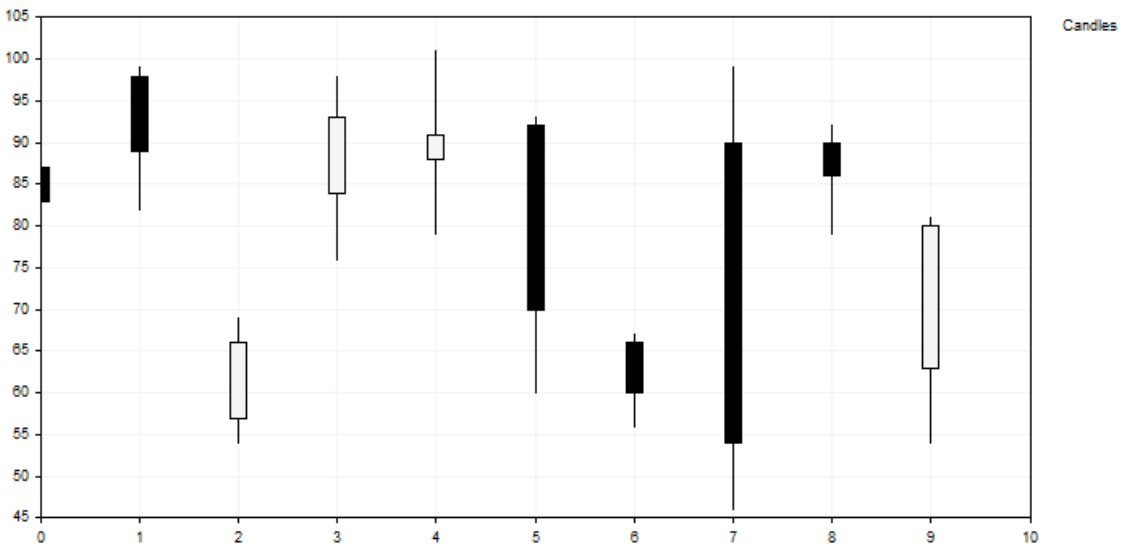
```
void  CustomPlotFunction(
   PlotFucntion  func      // fonksiyonun işaretçisi
   )
```

Parametreler

func

[in] Kullanıcı tanımlı eğri çizimi için kullanılan fonksiyonun işaretçisi

Örnek:



Çubuklardan olşan bu eğri aşağıdaki kod ile çizilmiştir:

```
//+------------------------------------------------------------------+
//|                                                CandleGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd.      |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| CCandle Sınıfı                                                   |
//| Kullanım: Mum çizimi için bir sınıf                              |
//+------------------------------------------------------------------+
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);     }
  };
//+------------------------------------------------------------------+
//| Constructor                                                      |
//+------------------------------------------------------------------+
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)
  {
  }
//+------------------------------------------------------------------+
//| Destructor                                                       |
//+------------------------------------------------------------------+
CCandle::~CCandle(void)
  {
  }
//+------------------------------------------------------------------+
//| Mum çizimi için kullanıcı tanımlı yöntem                         |
//+------------------------------------------------------------------+
void PlotCandles(double &x[],double &y[],int size,CGraphic *graphic,CCanvas *canvas,vo
  {
//--- nesneyi kontrol et
   CArrayObj *candles=dynamic_cast<CArrayObj*>(cbdata);
   if(candles==NULL || candles.Total()!=size)
      return;
//--- mumları çiz
   for(int i=0; i<size; i++)
     {
      CCandle *candle=dynamic_cast<CCandle*>(candles.At(i));
```

```
      if(candle==NULL)
         return;
      //--- ön hesaplama
      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
      //--- mumu çiz
      canvas.LineVertical(xc,high,low,0x000000);
      //--- mum gövdesini çiz
      canvas.FillRectangle(xc+width_2,open,xc-width_2,close,clr);
      canvas.Rectangle(xc+width_2,open,xc-width_2,close,0x000000);
     }
  }
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
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;
//--- değerleri oluştur
   for(int i=0; i<count; i++)
     {
      x[i] = i;
      y[i] = i;
      //--- değerleri hesapla
      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);
      //--- max ve min değerlerini bul
      if(i==0 || max<high)
         max=high;
      if(i==0 || min>low)
         min=low;
      //--- mumu oluştur
      CCandle *candle=new CCandle(open,close,high,low,width);
      candles.Add(candle);
     }
//--- grafiği oluştur
   CGraphic graphic;
   if(!graphic.Create(0,"CandleGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"CandleGraphic");
     }
//--- eğri oluştur
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_CUSTOM,"Candles");
//--- eğri özelliklerini ayarlar
   curve.CustomPlotFunction(PlotCandles);
   curve.CustomPlotCBData(GetPointer(candles));
```

```
//--- grafik özelliklerini ayarlar
   graphic.YAxis().Max((int)max);
   graphic.YAxis().Min((int)min);
//--- çiz
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

PointsType (Get yöntemi)

Noktalı eğri çizerken kullanılan noktaların biçim bayrağını alır.

```
ENUM_POINT_TYPE  PointsType()
```

Dönüş Değeri

Noktalı eğri çizerken kullanılan noktaların biçim bayrağının değeri.

PointsType (Set yöntemi)

Noktalı eğri çizerken kullanılan noktaların biçim bayrağını ayarlar.

```
void  PointsType(
   ENUM_POINT_TYPE  type      // bayrak değeri
   )
```

Parametreler

type

[in] Noktalı eğri çizerken kullanılan noktaların biçim bayrağının değeri.

StepsDimension (Get yöntemi)

Adım-tipli eğri düzleştirmede kullanılan boyut değerini alır.

```
int  StepsDimension()
```

Dönüş Değeri

Adım-tipli eğri düzleştirmede kullanılan boyut değeri.

StepsDimension (Set yöntemi)

Adım-tipli eğri düzleştirmede kullanılan boyut değerini ayarlar.

```
void  StepsDimension(
   const int  dimension      // boyut
   )
```

Parametreler

dimension

[in] Boyut (0 veya 1).

Not

0 — x (yatay çizgi dikeyden sonra gelir).

1 — y (dikey çizgi yataydan sonra gelir).

TrendLineCoefficients (Get yöntemi)

Diziye yazmak için trend çizgisi oranlarını alır.

```
double&  TrendLineCoefficients()
```

Dönüş Değeri

Trend çizgisi oranları.

TrendLineCoefficients (Set yöntemi)

Diziye yazmak için trend çizgisi oranlarını ayarlar.

```
void  TrendLineCoefficients(
   double&  coefficients[]      // oranların yazılacağı dizi
   )
```

Parametreler

coefficients[]

[out] Oranların yazılacağı dizi.

TrendLineColor (Get yöntemi)

Eğri için kullanılacak trend çizgisinin rengini alır.

```
uint  TrendLineColor()
```

Dönüş Değeri

Trend çizgisinin rengi.

TrendLineColor (Set yöntemi)

Eğri için kullanılacak trend çizgisinin rengini ayarlar.

```
void  TrendLineColor(
   const uint  clr      // trend çizgisinin rengi
   )
```

Parametreler

clr

[in] Çizgi rengi

TrendLineVisible (Get yöntemi)

Trend çizgisi görünürlük bayrağını alır.

```
bool  TrendLineVisible()
```

Dönüş Değeri

Trend çizgisi görünürlük bayrağının değeri.

TrendLineVisible (Set yöntemi)

Trend çizgisi görünürlük bayrağını ayarlar.

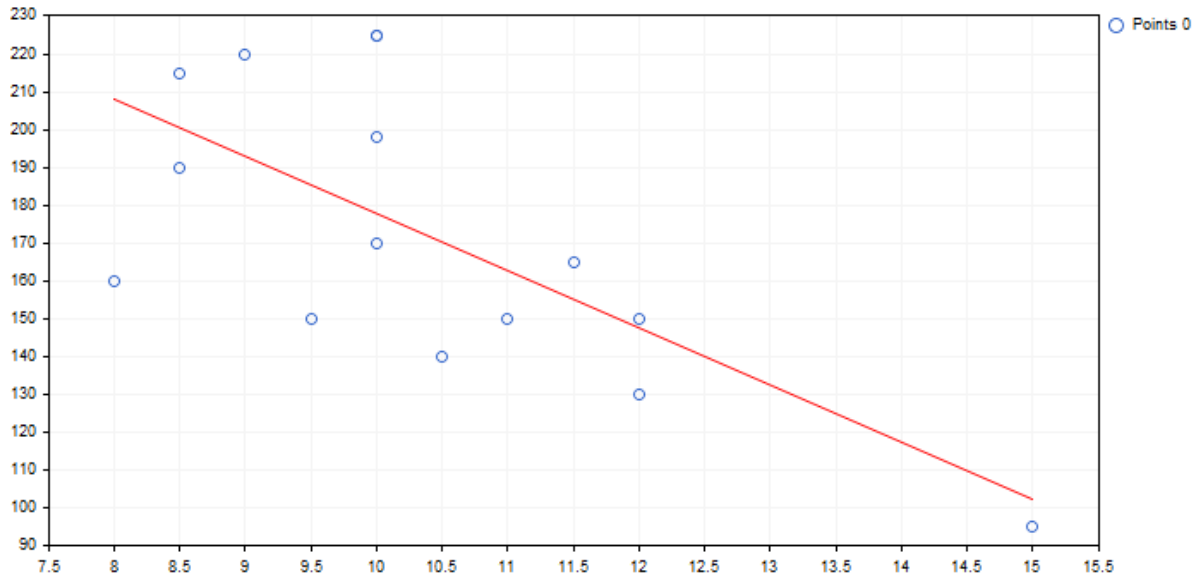
```
void  TrendLineVisible(
   const bool  visible      // bayrak değeri
   )
```

Parametreler

visible

[in] Trend çizgisi görünürlük bayrağının değeri.

Örnek:



Bahsedilen trend çizgisinin kodu ve çizelgeye çizilmesi:

```
//+------------------------------------------------------------------+
//|                                            TrendLineGraphic.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                            https://www.mql5.com |
//+------------------------------------------------------------------+
#include <Graphics\Graphic.mqh>
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
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
//--- grafiği oluştur
   CGraphic graphic;
   if(!graphic.Create(0,"TrendLineGraphic",0,30,30,780,380))
     {
      graphic.Attach(0,"TrendLineGraphic");
     }
//--- eğriyi oluştur
   CCurve *curve=graphic.CurveAdd(x,y,CURVE_POINTS);
//--- eğri özelliklerini belirt
   curve.TrendLineVisible(true);
   curve.TrendLineColor(ColorToARGB(clrRed));
//--- çiz
   graphic.CurvePlotAll();
   graphic.Update();
  }
```

Update

Eğri koordinatlarını ayarlar.

Y koordinatları ile çalışılana versiyon. Geçirilen dizinin indisleri X koordinatları olarak alınır.

```
void  Update(
   const double&  y[]   // Y koordinatları
   )
```

X ve Y koordinatları ile kullanılan versiyon.

```
void  Update(
   const double&  x[],     // X koordinatları
   const double&  y[]      // Y koordinatları
   )
```

CPoint2D noktalarıyla kullanılan versiyon.

```
void  Update(
   const CPoint2D&  points[]      // Eğri koordinatları
   )
```

CurveFunction fonksiyonunun işaretçisi ile çalışmak için kullanılan versiyon.

```
void  Update(
   CurveFunction  function,     // eğriyi tanımlayan fonksiyonun işaretçisi
   const double   from,         // fonksiyon argümanının başlangıç değeri
   const double   to,           // fonksiyon argümanının son değeri
   const double   step          // argüman artış değeri
   )
```

Parametreler

x[]

[in] X koordinatları.

y[]

[in] Y koordinatları.

points[]

[in] Eğri koordinatları.

function

[in] Eğriyi tanımlayan fonksiyonun işaretçisi

from

[in] Fonksiyon argümanının başlangıç değeri

to

[in] Fonksiyon argümanının başlangıç değeri

step

[in] Argüman artışı

Visible (Get yöntemi)

Fonksiyonun çizelge üzerindeki görünürlüğünü ayarlayan bayrağı alır.

```
void  Visible(
   const bool  visible      //
   )
```

Dönüş Değeri

Fonksiyonun çizelge üzerindeki görünürlüğünü ayarlayan bayrağın değeri.

Visible (Set yöntemi)

Fonksiyonun çizelge üzerindeki görünürlüğünü ayarlayan bayrağı ayarlar.

```
void  Visible(
   const bool  visible      // bayrak değeri
   )
```

Parametreler

visible

[in] Fonksiyonun çizelge üzerindeki görünürlüğünü ayarlayan bayrağın değeri.

CGraphic

CGraphic, özel çizelgeler oluşturmak için temel sınıftır.

Açıklama

CGraphic sınıfı özel grafiklerle çalışmak için sayısız araçlar içerir.

Bu sınıfta ana çizelge elemanları depolanır, bunların parametreleri ayarlanır ve çizim gerçekleştirilir.

Ayrıca, çizelgede kullanılacak eğrileri depolar ve çeşitli görüntüleme seçenekleri sunar.

Bildirim

```
class CGraphic
```

Başlık

```
#include <Graphics\Graphic.mqh>
```

Sınıf yöntemleri

Yöntem	Açıklama
Create	Bir çizelge nesnesine tutturulmuş grafiksel kaynak oluşturur
Destroy	Çizelgeyi kaldırır ve grafiksel kaynağı yok eder
Update	Uygulanmış değişimleri görüntüler
ChartObjectName	Çizelgeye tutturulmuş bir nesnenin ismine dönüş yapar
ResourceName	Grafiksel kaynağın ismine dönüş yapar
XAxis	X ekseninin işaretçisine dönüş yapar
YAxis	Y ekseninin işaretçisine dönüş yapar
GapSize	Çizelge elemanları arasındaki boşluk miktarını alır/ayarlar
BackgroundColor	Arka-plan rengini alır/ayarlar
BackgroundMain	Çizelge başlığını alır/ayarlar
BackgroundMainSize	Çizelge başlığının yazı tipi boyutunu alır/ayarlar
BackgroundMainColor	Çizelge başlığının rengini alır/ayarlar
BackgroundSub	Alt başlığı alır/ayarlar
BackgroundSubSize	Alt başlığın yazı tipi boyutunu alır/ayarlar
BackgroundSubColor	Çizelge alt başlığının rengini alır/ayarlar

Yöntem	Açıklama
GridLineColor	Izgara çizgisinin rengini alır/ayarlar
GridBackgroundColor	Izgara arkaplan rengini alır/ayarlar
GridCircleRadius	Ağ düğümü noktalarının yarıçapını alır/ayarlar
GridCircleColor	Ağ düğümü noktalarının rengini alır/ayarlar
GridHasCircle	Ağ düğümü noktalarının çizim bayrağını alır/ayarlar
GridAxisLineColor	Reel çizelge ekseninin renk değerini alır.
HistoryNameWidth	Eğri ismi için izin verilen maksimum görüntüleme genişliğini alır/ayarlar
HistoryNameSize	Eğri isminin yazı tipi boyutunu alır/ayarlar
HistorySymbolSize	Noktalama işaretlerinin boyutunu alır/ayarlar
TextAdd	Çizelgeye bir metin ekler
LineAdd	Çizelgeye bir çizgi ekler
CurveAdd	Bir eğri oluşturup çizelgeye ekler
CurvePlot	Daha önce oluşturulmuş bir eğriyi indisine göre çizer
CurvePlotAll	Daha önce oluşturulmuş tüm eğrileri çizer
CurveGetByIndex	Belirtilen indise karşılık gelen eğriye dönüş yapar
CurveGetByName	Belirtilen isimdeki eğriye dönüş yapar
CurveRemoveByIndex	Belirtilen indisteki eğriyi siler.
CurveRemoveByName	Belirtilen isimli eğriyi siler.
CurvesTotal	Belirtilen çizelgedeki eğri sayısını alır.
MarksToAxisAdd	Belirtilen çizelge eksenine ölçek işareti ekler.
MajorMarkSize	Belirtilen çizelge eksenindeki ölçek işaretlerinin boyutunu alır/ayarlar.
FontSet	Mevcut yazı-tipi parametrelerini ayarlar
FontGet	Mevcut yazı-tipi parametrelerini alır
Attach	Bir grafiksel kaynak alır/ayarlar ve CGraphic sınıf örneğine tutturur
CalculateMaxMinValues	CHer iki eksen için minimum ve maksimum değerleri hesapla.
Height	Çizelge yüksekliğini piksel cinsinden al.
IndentDown	Alt kenara göre çizelge girintisi değerini alır/ayarlar.
IndentLeft	Sol kenara göre çizelge girintisi değerini alır/ayarlar.

Yöntem	Açıklama
IndentRight	Sağ kenara göre çizelge girintisi değerini alır/ayarlar.
IndentUp	Üst kenara göre çizelge girintisi değerini alır/ayarlar.
Redraw	Çizelgeyi yeniden çizer.
ResetParameters	Yeniden çizim parametrelerini sıfırlar.
ScaleX	Değeri X eksenine göre ölçekle.
ScaleY	Değeri Y eksenine göre ölçekle.
SetDefaultParameters	Çizelge parametrelerini varsayılan olarak ayarla.
Width	Çizelge genişliğini piksel cinsinden alır.

Create

Çizelge tutturulmuş bir grafiksel nesne oluşturur

```
bool  Create(
   const long     chart,      // çizelge tanımlayıcısı
   const string   name,       // isim
   const int      subwin,     // alt-pencere indisi
   const int      x1,         // x1 koordinatı
   const int      y1,         // y1 koordinatı
   const int      x2,         // x2 koordinatı
   const int      y2          // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] İsim.

subwin

[in] Alt-pencere indisi.

x1

[in] X1 koordinatı.

y1

[in] Y1 koordinatı.

x2

[in] X2 koordinatı.

y2

[in] Y2 koordinatı.

Destroy

Bir çizelgeyi siler ve grafiksel kaynağı yok eder.

```
void  Destroy()
```

Update

Uygulanmış değişimleri görüntüler.

```
void  Update(
   const bool  redraw=true      // bayrak
   )
```

Parametreler

redraw=true

[in] Bayrak değeri

ChartObjectName

Çizelgeye tutturulmuş bir nesnenin ismine dönüş yapar.

```
string  ChartObjectName()
```

Dönüş Değeri

Çizelgeye tutturulmuş bir nesnenin ismi.

ResourceName

Bir grafiksel kaynağın isim bilgisini alır.

```
string  ResourceName()
```

Dönüş Değeri

Grafiksel kaynağın ismi.

XAxis

X ekseninin işaretçisine dönüş yapar.

```
CAxis  *XAxis()
```

Dönüş Değeri

X ekseninin işaretçisi.

YAxis

Y ekseninin işaretçisine dönüş yapar.

```
CAxis  *YAxis()
```

Dönüş Değeri

Y ekseninin işaretçisi.

GapSize (Get yöntemi)

Çizelge elemanları arasındaki boşluk miktarına dönüş yapar.

```
int  GapSize()
```

Dönüş Değeri

Piksel cinsinden boyut.

GapSize (Set yöntemi)

Çizelge elemanları arasındaki boşluk miktarını ayarlar

```
void  GapSize(
   const int  size     // boşluk miktarı
   )
```

Parametreler

size

[in] Piksel cinsinden boşluk miktarı.

BackgroundColor (Get yöntemi)

Arkaplan rengine dönüş yapar.

```
color  BackgroundColor()
```

BackgroundColor (Set yöntemi)

Arka-plan rengini ayarlar.

```
void  BackgroundColor(
   const color  clr      // arkaplan rengi
   )
```

Parametreler

clr

[in] Arkaplan rengi.

BackgroundMain (Get yöntemi)

Çizelge başlığına dönüş yapar

```
string  BackgroundMain()
```

BackgroundMain (Set yöntemi)

Çizelge başlığının metnini ayarlar.

```
void  BackgroundMain(
   const string  main      // başlık metni
   )
```

Parametreler

main

[in] Çizelge başlığının metni

BackgroundMainSize (Get yöntemi)

Başlık boyutuna dönüş yapar.

```
int  BackgroundMainSize()
```

Dönüş Değeri

Başlığın yazı tipi boyutu.

BackgroundMainSize (Set yöntemi)

Başlığın yazı tipi boyutunu ayarlar.

```
void  BackgroundMainSize(
   const int  size     // başlık boyutu
   )
```

Parametreler

size

[in] Başlığın yazı tipi boyutu.

BackgroundMainColor (Get yöntemi)

Başlık rengine dönüş yapar.

```
color  BackgroundMainColor()
```

Dönüş Değeri

Başlık rengi.

BackgroundMainColor (Set yöntemi)

Başlık rengini ayarlar.

```
void  BackgroundMainColor(
   const color  clr      // başlık rengi
   )
```

Parametreler

clr

[in] Başlık rengi.

BackgroundSub (Get yöntemi)

Alt başlığa dönüş yapar.

```
string  BackgroundSub()
```

Dönüş Değeri

Alt başlık metni.

BackgroundSub (Set yöntemi)

Alt başlık metnini ayarlar.

```
void  BackgroundSub (Set yöntemi)(
   const string  sub      // alt başlık metni
   )
```

Parametreler

sub

[in] Alt başlık metni.

BackgroundSubSize (Get yöntemi)

Alt başlığın yazı tipi boyutuna dönüş yapar

```
int BackgroundSubSize()
```

BackgroundSubSize (Get yöntemi)

Alt başlık boyutunu ayarlar.

```
void  BackgroundSubSize(
   const int  size       // alt başlığın yazı tipi boyutu
   )
```

Parametreler

size

[in] Alt başlığın yazı tipi boyutu

BackgroundSubColor (Get yöntemi)

Alt başlık rengine dönüş yapar.

```
color  BackgroundSubColor()
```

BackgroundSubColor (Set yöntemi)

Alt başlık rengini ayarlar.

```
void  BackgroundSubColor(
   const color  clr      // alt başlık rengi
   )
```

Parametreler

clr

[in] Alt başlık rengi.

GridLineColor (Get yöntemi)

Izgara çizgi rengine dönüş yapar

```
color  GridLineColor()
```

Dönüş Değeri

Izgara çizgisinin rengi.

GridLineColor (Set yöntemi)

Izgara çizgi rengini ayarlar.

```
void  GridLineColor(
   const color  clr      // çizgi rengi
   )
```

Parametreler

clr

[in] Izgara çizgi rengi.

GridBackgroundColor (Get yöntemi)

Izgara arkaplan rengine dönüş yapar

```
color  GridBackgroundColor()
```

Dönüş Değeri

Izgara arkaplan rengi

GridBackgroundColor (Set yöntemi)

Izgara arkaplan rengini ayarlar

```
void  GridBackgroundColor(
   const color  clr      // ızgara arkaplan rengi
   )
```

Parametreler

clr

[in] Izgara arkaplan rengi

GridCircleRadius (Get yöntemi)

Ağ düğümü noktalarının yarı çapına dönüş yapar.

```
int  GridCircleRadius()
```

Dönüş Değeri

Nokta yarı çapının piksel cinsinden değeri.

GridCircleRadius (Set yöntemi)

Ağ düğümü noktalarının yarıçapını ayarlar.

```
void  GridCircleRadius(
   const int  r      // yarıçap
   )
```

Parametreler

r

[in] Piksel cinsinden nokta yarı çapı.

GridCircleColor (Get yöntemi)

Ağ düğümü noktalarının rengine dönüş yapar.

```
color  GridCircleColor()
```

Dönüş Değeri

Nokta rengi.

GridCircleColor (Set yöntemi)

Ağ düğümü noktalarının rengini ayarlar.

```
void  GridCircleColor(
   const color  clr      // nokta rengi
   )
```

Parametreler

clr

[in] Nokta rengi.

GridHasCircle (Get yöntemi)

Izgara düğümlerinindeki noktaların görüntülenip görüntülenmeyeceğini belirten bayrağa dönüş yapar.

```
bool  GridHasCircle()
```

Dönüş Değeri

Bayrak değeri.

Not

true — noktaları görüntüle

false — noktaları görüntüleme

GridHasCircle (Set yöntemi)

Izgara düğümlerinindeki noktaların görüntülenip görüntülenmeyeceğini belirten bayrağı ayarlar.

```
void  GridHasCircle(
   const bool  has
   )
```

Parametreler

has

[in] Bayrak değeri.

Not

true — noktaları görüntüle

false — noktaları görüntüleme

GridAxisLineColor (Get yöntemi)

Reel çizelge ekseninin renk değerini alır.

```
uint  GridAxisLineColor()
```

Dönüş Değeri

Reel çizelge ekseninin rengi.

GridAxisLineColor (Set yöntemi)

Reel çizelge ekseninin renk değerini ayarlar.

```
void  GridAxisLineColor(
   const uint  clr      // çizelge ekseninin renk değeri
   )
```

Parametreler

clr

[in] Reel çizelge ekseninin renk değeri.

HistoryNameWidth (Get yöntemi)

Eğri ismi için izin verilen maksimum görüntüleme genişliğine dönüş yapar.

```
int  HistoryNameWidth()
```

Dönüş Değeri

Piksel cinsinden makksimum genişlik.

Not

Eğri ismi izin verilen maksimum genişliği aşarsa, isim kısa kesilir ve sonuna noktalar eklenir.

HistoryNameWidth (Set yöntemi)

Eğri ismi için izin verilen maksimum görüntüleme genişliği ayarlar.

```
void  HistoryNameWidth(
   const int  width      // maksimum genişlik
   )
```

Parametreler

width

[in] Piksel cinsinden maksimum genişlik.

Not

Eğri ismi izin verilen maksimum genişliği aşarsa, isim kısa kesilir ve sonuna noktalar eklenir.

HistoryNameSize (Get yöntemi)

Eğri isminin yazı tipi boyutuna dönüş yapar.

```
int  HistoryNameSize()
```

Dönüş Değeri

Eğri isminin yazı tipi boyutu.

HistoryNameSize (Set yöntemi)

Eğri isminin yazı tipi boyutunu ayarlar.

```
void  HistoryNameSize (Set yöntemi)(
   const int  size      // ismin yazı tipi boyutu
   )
```

Parametreler

size

[in] İsmin yazı tipi boyutu.

HistorySymbolSize (Get yöntemi)

Çizelgedeki noktalama işaretlerinin boyutuna dönüş yapar

```
int  HistorySymbolSize()
```

Dönüş Değeri

Noktalama işaretlerinin boyutu

HistorySymbolSize (Set yöntemi)

Çizelgedeki noktalama işaretlerinin boyutunu ayarlar

```
void  HistorySymbolSize(
   const int  size      // sembol boyutu
   )
```

Parametreler

size

[in] Noktalama işaretlerinin boyutu.

TextAdd

Çizelgeye bir metin ekler.

Bu versiyonda X ve Y koordinatları kullanılır

```
void  TextAdd(
   const int     x,              // X koordinatı
   const int     y,              // Y koordinatı
   const string  text,           // metin
   const uint    clr,            // renk
   const uint    alignment=0     // hizalama
   )
```

CPoint için kullanılan versiyon

```
void  TextAdd(
   const CPoint  &point,         // nokta koordinatı
   const string  text,           // metin
   const uint    clr,            // renk
   const uint    alignment=0     // hizalama
   )
```

Parametreler

x

[in] X koordinatı.

y

[in] Y koordinatı.

&point

[in] Nokta koordinatı.

text

[in] Metin.

clr

[in] Renk.

alignment=0

[in] Hizalama.

LineAdd

Çizelgeye bir çizgi ekler.

Bu versiyonda X ve Y koordinatları kullanılır

```
void  LineAdd(
   const int   x1,         // x1 koordinatı
   const int   y1,         // y1 koordinatı
   const int   x2,         // x2 koordinatı
   const int   y2,         // y2 koordinatı
   const uint  clr,        // renk
   const uint  style       // stil
   )
```

CPoint için kullanılan versiyon

```
void  LineAdd2(
   const CPoint  &point1,     // ilk noktanın koordinatı
   const CPoint  &point2,     // ikinci noktanın koordinatı
   const uint    clr,         // renk
   const uint    style        // stil
   )
```

Parametreler

x1

[in] X1 koordinatı.

y1

[in] Y1 koordinatı.

x2

[in] X2 koordinatı.

y2

[in] Y2 koordinatı.

&point1

[in] İlk noktanın koordinatı.

&point2

[in] İkinci noktanın koordinatı.

clr

[in] Renk.

style

[in] Stil.

CurveAdd

Bir eğri oluşturup çizelgeye ekler.

Bu versiyon Y koordinatlarını kullanır (eğri rengi otomatik olarak belirlenir)

```
CCurve* CurveAdd(
   const double      &y[],          // Y koordinatları
   ENUM_CURVE_TYPE   type,          // eğri tipi
   const string      name=NULL      // eğri ismi
   )
```

Not

Y dizisinin indisleri eğri çizimi sırasında X koordinatları olarak kullanılır.

Bu versiyon X veY koordinatlarını kullanır (eğri rengi otomatik olarak belirlenir)

```
CCurve*  CurveAdd(
   const double      &x[],          // Y koordinatları
   const double      &y[],          // Y koordinatları
   ENUM_CURVE_TYPE   type,          // eğri tipi
   const string      name=NULL      // eğri ismi
   )
```

CPoint2D noktalarıyla eğri çizimi için kullanılan versiyon (eğri rengi otomatik olarak belirlenir)

```
CCurve*  CurveAdd(
   const CPoint2D    &points[],     // nokta koordinatları
   ENUM_CURVE_TYPE   type,          // eğri tipi
   const string      name=NULL      // eğri ismi
   )
```

CurveFunction fonksiyonunun işaretçisi ile eğri çizimi yapılan versiyon (eğri rengi otomatik olarak belirlenir)

```
CCurve*  CurveAdd(
   CurveFunction     function,      // fonksiyon işaretçisi
   const double      from,          // argümanın başlangıç değeri
   const double      to,            // argümanın son değeri
   const double      step,          // argümanın artış değeri
   ENUM_CURVE_TYPE   type,          // eğri tipi
   const string      name=NULL      // eğri ismi
   )
```

Bu versiyon Y koordinatlarını kullanır (eğri rengini kullanıcı belirler)

```
CCurve*  CurveAdd(
   const double      &y[],          // Y koordinatları
   const uint        clr,           // eğri rengi
   ENUM_CURVE_TYPE   type,          // eğri tipi
   const string      name=NULL      // eğri ismi
   )
```

Not

Y dizisinin indisleri eğri çizimi sırasında X koordinatları olarak kullanılır.

Bu versiyon X veY koordinatlarını kullanır (eğri rengini kullanıcı belirler)

```
CCurve*  CurveAdd(
   const double      &x[],          // Y koordinatları
   const double      &y[],          // Y koordinatları
   const uint        clr,           // eğri rengi
   ENUM_CURVE_TYPE   type,          // eğri tipi
   const string      name=NULL      // eğri ismi
   )
```

CPoint2D noktalarıy eğri çizimi için kullanılan versiyon (eğri rengini kullanıcı belirler)

```
CCurve*  CurveAdd(
   const CPoint2D    &points[],     // nokta koordinatları
   const uint        clr,           // eğri rengi
   ENUM_CURVE_TYPE   type,          // eğri tipi
   const string      name=NULL      // eğri ismi
   )
```

CurveFunction fonksiyonunun işaretçisi ile eğri çizimi yapılan versiyon (eğri rengini kullanıcı belirler)

```
CCurve*  CurveAdd(
   CurveFunction     function,      // fonksiyon işaretçisi
   const double      from,          // argümanın başlangıç değeri
   const double      to,            // argümanın son değeri
   const double      step,          // argümanın artış değeri
   const uint        clr,           // eğri rengi
   ENUM_CURVE_TYPE   type,          // eğri tipi
   const string      name=NULL      // eğri ismi
   )
```

Parametreler

&x[]

[in] X koordinatı.

&y[]

[in] Y koordinatı.

&points[]

[in] Noktaların koordinatları.

function

[in] Fonksiyonun işaretçisi.

from

[in] Argümanın başlangıç değeri.

to

[in] Argümanın son değeri.

step

[in] Argümanın artış değeri.

type

[in] Eğri tipi.

name=NULL

[in] Eğri ismi.

clr

[in] Eğri rengi.

Dönüş Değeri

Oluşturulan eğrinin işaretçisi.

CurvePlot

Daha önce oluşturulmuş bir eğriyi indisine göre çizer.

```
bool  CurvePlot(
   const int  index      // indis
   )
```

Parametreler

index

[in] Eğri indisi

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'

CurvePlotAll

Çizelgeye eklenmiş olan eğrilerin tamamını çizer.

```
bool  CurvePlotAll()
```

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'

CurveGetByIndex

Belirtilen indise karşılık gelen eğriye dönüş yapar.

```
CCurve*  CurveGetByIndex(
   const int  index      // eğri indisi
   )
```

Parametreler

index

[in] Eğri indisi.

Dönüş Değeri

Belirtilen indise karşılık gelen eğri.

CurveGetByName

Belirtilen isimdeki eğriye dönüş yapar.

```
CCurve*  CurveGetByName(
   const string  name      // eğri ismi
   )
```

Parametreler

name

[in] Eğri ismi.

Dönüş Değeri

Belirtilen isimle bulunan ilk eğrinin işaretçisi.

CurveRemoveByIndex

Belirtilen indisteki eğriyi siler.

```
bool  CurveRemoveByIndex(
   const int  index      // eğri indisi
   )
```

Parametreler

index

[in] Silinecek eğrinin indisi.

Dönüş Değeri

true — başarılı, aksi durumda — false.

CurveRemoveByName

Belirtilen isimli eğriyi siler.

```
bool  CurveRemoveByName(
   const string  name      // eğri ismi
   )
```

Parametreler

name

[in] Silinecek eğrinin ismi.

Dönüş Değeri

true — başarılı, aksi durumda — false.

CurvesTotal

Belirtilen çizelgedeki eğri sayısını alır.

```
int  CurvesTotal()
```

Dönüş Değeri

Eğrilerin sayısı.

Not

Çizim ve görünürlük stillerine bakılmaksızın tüm eğriler hesaba katılır.

MarksToAxisAdd

Belirtilen çizelge eksenine ölçek işareti (tik) ekler.

```
bool  MarksToAxisAdd(
   const double         &marks[],        // tik koordinatları
   const int            mark_size,       // tik genişliği
   ENUM_MARK_POSITION   position,        // tik konumu
   const int            dimension=0      // boyut
   )
```

Parametreler

&marks[]

[in] Tik koordinatları

mark_size

[in] Tik genişliği

position

[in] Tik konumu

dimension=0

[in] 0 — X ksenine eklenir,

1 — Y eksenine eklenir

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'

MajorMarkSize (Get yöntemi)

Koordinat eksenlerindeki ölçeğin tik genişliğine dönüş yapar.

```
int  MajorMarkSize()
```

MajorMarkSize (Set yöntemi)

Koordinat eksenlerindeki ölçeğin tik genişliğini ayarlar.

```
void  MajorMarkSize(
   const int  size      // tik genişliği
   )
```

Parametreler

size

[in] Piksel cinsinden tik genişliği.

FontSet

Mevcut font parametrelerini ayarlar.

```
bool  FontSet(
   const string  name,        // isim
   const int     size,        // boyut
   const uint    flags=0,     // bayraklar
   const uint    angle=0      // açı
   )
```

Parametreler

name

[in] İsim.

size

[in] Boyut.

flags=0

[in] Bayraklar.

angle=0

[in] Açı.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'

FontGet

Mevcut font parametrelerrini alır.

```
void  FontGet(
   string  &name,      // isim
   int     &size,      // boyut
   uint    &flags,     // bayraklar
   uint    &angle      // açı
   )
```

Parametreler

&name

[out] İsim.

&size

[out] Boyut.

&flags

[out] Bayraklar.

&angle

[out] Açı.

Attach

OBJ_BITMAP_LABEL nesnesinden grafiksel kaynak alıp CGraphic sınıf örneğine tutturma versiyonu:

```
bool  Attach(
   const long    chart_id,     // çizelge tanıtıcısı
   const string  objname       // grafiksel kaynak ismi
   )
```

OBJ_BITMAP_LABEL nesnesi için grafiksel kaynak oluşturup CGraphic sınıf örneğine tutturma versiyonu:

```
bool  Attach(
   const long    chart_id,     // çizelge tanıtıcısı
   const string  objname,      // grafiksel kaynak ismi
   const int     width,        // görüntü genişliği
   const int     height        // görüntü yüksekliği
   )
```

Parametreler

chart_id

[in] Çizelge tanımlayıcısı.

objname

[in] Grafiksel nesnenin ismi.

width

[in] Kaynaktaki görüntü genişliği.

height

[in] Kaynaktaki görüntü yüksekliği.

Dönüş Değeri

true — başarılı, false — nesne tutturulamadı.

CalculateMaxMinValues

CHer iki eksen için minimum ve maksimum değerleri hesapla.

```
void  CalculateMaxMinValues()
```

Height

Çizelge yüksekliğini piksel cinsinden al.

```
int  Height()
```

Dönüş Değeri

Piksel cinsinden çizelge yüksekliği.

IndentDown (Get yöntemi)

Alt kenara göre çizelge girintisi değerini alır.

```
int  IndentDown()
```

Dönüş Değeri

Piksel cinsinden girinti değeri.

IndentDown (Set yöntemi)

Alt kenara göre çizelge girintisi değerini ayarlar.

```
void  IndentDown(
   const int  down      // girinti miktarı
   )
```

Parametreler

down

[in] Piksel cinsinden girinti değeri.

IndentLeft (Get yöntemi)

Sol kenara göre çizelge girintisi değerini alır.

```
int  IndentLeft()
```

Dönüş Değeri

Piksel cinsinden girinti değeri.

IndentLeft (Set yöntemi)

Sol kenara göre çizelge girintisi değerini ayarlar.

```
void  IndentLeft(
   const int  left      // girinti miktarı
   )
```

Parametreler

left

[in] Piksel cinsinden girinti değeri.

IndentRight (Get yöntemi)

Sağ kenara göre çizelge girintisi değerini alır.

```
int  IndentRight()
```

Dönüş Değeri

Piksel cinsinden girinti değeri.

IndentRight (Set yöntemi)

Sağ kenara göre çizelge girintisi değerini ayarlar.

```
void  IndentRight(
   const int  right      // girinti miktarı
   )
```

Parametreler

right

[in] Piksel cinsinden girinti değeri.

IndentUp (Get yöntemi)

Üst kenara göre çizelge girintisi değerini alır.

```
int  IndentUp()
```

Dönüş Değeri

Piksel cinsinden girinti değeri.

IndentUp (Set yöntemi)

Üst kenara göre çizelge girintisi değerini alır.

```
void  IndentUp(
   const int  up      // girinti miktarı
   )
```

Parametreler

up

[in] Piksel cinsinden girinti değeri.

Redraw

Çizelgeyi yeniden çizer.

```
bool  Redraw(
   const bool  rescale=false      // bayrak değeri
   )
```

Parametreler

rescale=false

[in] Çizelgenin yeniden çizilip çizilmeyeceğini gösteren bayrak.

Dönüş Değeri

Başarılı ise 'true', aksi durumda 'false'

ResetParameters

Yeniden çizim parametrelerini sıfırlar.

```
void  ResetParameters()
```

ScaleX

Değeri X eksenine göre ölçekle.

```
virtual int  ScaleX(
   double  x      // X eksenine göre değer
   )
```

Parametreler

x

[in] X eksenine göre reel değer.

Dönüş Değeri

Piksel cinsinden bir değer.

Not

Çizelgede görüntülemek için bir reel değer piksel cinsine dönüştürülür.

ScaleY

Değeri Y eksenine göre ölçekle.

```
virtual int  ScaleY(
   double  y      // Y eksenine göre değer
   )
```

Parametreler

y

[in] Y eksenine göre reel değer.

Dönüş Değeri

Piksel cinsinden bir değer.

Not

Çizelgede görüntülemek için bir reel değer piksel cinsine dönüştürülür.

SetDefaultParameters

Çizelge parametrelerini varsayılan olarak ayarla.

```
void  SetDefaultParameters()
```

Width

Çizelge genişliğini piksel cinsinden alır.

```
int  Width()
```

Dönüş Değeri

Piksel cinsinden çizelge genişliği.

Teknik göstergeler ve zaman-serileri

Bu bölüm, teknik göstergeler ve zaman-serileri sınıfları ile çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için gereken açıklamaları içermektedir.

Teknik göstergeler ve zaman-serileri sınıfları, uygulamalar (betikler, Uzman Danışmanlar) geliştirirken zaman kazandıracaktır.

MQL5 Standart Kütüphanesinin gösterge ve zaman-serileri sınıfları, terminalin Include\Indicators çalışma klasöründe yer almaktadır.

Sınıf/grup	Açıklama
Temel sınıflar	Temel ve yardımcı sınıfların grubu
Zaman-serisi sınıfları	Zaman-serisi sınıflarının grubu
Trend Göstergeleri	Trend göstergesi sınıflarının grubu
Osilatörler	Osilatör gösterge sınıflarının grubu
Hacim göstergeleri	Hacim göstergesi sınıflarının grubu
Bill Williams Göstergeleri	Bill Williams göstergelerinin sınıfları
Özel göstergeler	Özel gösterge sınıfı

Teknik Göstergeler ve Zaman-serileri için Temel ve Yardımcı Sınıflar

Bu bölüm, teknik göstergeler ve zaman-serileri için hazırlanmış temel ve yardımcı sınıflar ile çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için gereken açıklamaları içermektedir.

Sınıf/grup	Açıklama
CSpreadBuffer	Geçmiş spread (alış satış farkı) verileri tamponunun sınıfı
CTimeBuffer	Geçmiş açılış fiyatları tamponunun sınıfı
CTickVolumeBuffer	Geçmiş tik hacimleri tamponunun sınıfı
CRealVolumeBuffer	Geçmiş işlem hacimleri tamponunun sınıfı
CDoubleBuffer	double tipli veri tamponları için temel sınıf
COpenBuffer	Çubuk açılış fiyatları tamponunun sınıfı
CHighBuffer	Yüksek çubuk fiyatları tamponunun sınıfı
CLowBuffer	Düşük çubuk fiyatları tamponunun sınıfı
CCloseBuffer	Çubuk kapanış fiyatları tamponunun sınıfı
CIndicatorBuffer	Teknik gösterge tamponu sınıfı
CSeries	Zaman-serisi verilerine erişim için temel sınıf
CPriceSeries	Fiyat verilerine erişim için temel sınıf
CIndicator	Teknik göstregeler için temel sınıf
CIndicators	Teknik gösterge ve zaman-serileri koleksiyonu

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

CSpreadBuffer

CSpreadBuffer sınıfı, geçmiş çubukların spread (alış satış farkı) fiyatlarına kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CSpreadBuffer sınıfı, geçmiş çubukların spread fiyatlarına kolay erişim sağlar.

Bildirim

```
class CSpreadBuffer: public CArrayInt
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayInt
      CSpreadBuffer

Sınıf Yöntemleri

Özellikler	
Size	Tampon boyutunu ayarlar
Ayarlar	
SetSymbolPeriod	Sembol ve periyot değerlerini ayarlar
Veri Erişim Yöntemleri	
At	Tampon elemanını indis kullanarak alır
Veri Güncelleme Yöntemleri	
virtual Refresh	Tamponu günceller
virtual RefreshCurrent	Mevcut değeri günceller

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CArrayInt

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Size

Tampon boyutunu ayarlar.

```
void  Size(
   const int  size      // yeni boyut
   )
```

Parametreler

size

[in] Yeni tampon boyutu.

SetSymbolPeriod

Sembol ve periyot değerlerini ayarlar.

```
void  SetSymbolPeriod(
   const string           symbol,      // sembol
   const ENUM_TIMEFRAMES  period       // periyot
   )
```

Parametreler

symbol

[in] Yeni sembol.

period

[in] Yeni periyot değeri (ENUM_TIMEFRAMES sayımının değerlerinden biri).

At

Tampon elemanını indisine göre alır.

```
int  At(
   const int  index      // indis
   ) const
```

Parametreler

index

[in] Tampon elemanının indisi.

Dönüş değeri

Belirtilen indisteki tampon elemanı.

Refresh

Tamponu günceller.

```
virtual bool  Refresh()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

RefreshCurrent

Tamponun -sıfır indisli- mevcut değerini günceller.

```
virtual bool  RefreshCurrent()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

CTimeBuffer

CTimeBuffer sınıfı, geçmiş çubukların açılış zamanlarına kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CTimeBuffer sınıfı, geçmiş çubukların açılış zamanlarına erişim sağlar.

Bildirim

```
class CTimeBuffer: public CArrayLong
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayLong
      CTimeBuffer

Sınıf Yöntemleri

Özellikler	
Size	Tampon boyutunu ayarlar
Ayarlar	
SetSymbolPeriod	Sembol ve periyot değerlerini ayarlar
Veri Erişim Yöntemleri	
At	Tampon elemanını indis kullanarak alır
Veri Güncelleme Yöntemleri	
virtual Refresh	Tamponu günceller
virtual RefreshCurrent	Mevcut değeri günceller

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CArrayLong

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Size

Tampon boyutunu ayarlar.

```
void  Size(
   const int  size      // yeni boyut
   )
```

Parametreler

size

[in] Yeni tampon boyutu.

SetSymbolPeriod

Sembol ve periyot değerlerini ayarlar.

```
void  SetSymbolPeriod(
   const string           symbol,      // sembol
   const ENUM_TIMEFRAMES  period       // periyot
   )
```

Parametreler

symbol

[in] Yeni sembol.

period

[in] Yeni periyot değeri (ENUM_TIMEFRAMES sayımının değerlerinden biri).

At

Tampon elemanını indisine göre alır.

```
long  At(
   const int  index      // indis
   ) const
```

Parametreler

index

[in] Tampon elemanının indisi.

Dönüş değeri

Belirtilen indisteki tampon elemanı.

Refresh

Tamponu günceller.

```
virtual bool  Refresh()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

RefreshCurrent

Tamponun -sıfır indisli- mevcut değerini günceller.

```
virtual bool  RefreshCurrent()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

CTickVolumeBuffer

CTickVolumeBuffer sınıfı, geçmiş çubukların tik hacimlerine kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CTickVolumeBuffer sınıfı, geçmiş çubukların tik hacimlerine kolay sağlar.

Bildirim

```
class CTickVolumeBuffer: public CArrayLong
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayLong
      CTickVolumeBuffer

Sınıf Yöntemleri

Özellikler	
Size	Tampon boyutunu ayarlar
Ayarlar	
SetSymbolPeriod	Sembol ve periyot değerlerini ayarlar
Veri Erişim Yöntemleri	
At	Tampon elemanını indis kullanarak alır
Veri Güncelleme Yöntemleri	
virtual Refresh	Tamponu günceller
virtual RefreshCurrent	Mevcut değeri günceller

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CArrayLong

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Size

Tampon boyutunu ayarlar.

```
void  Size(
   const int  size      // yeni boyut
   )
```

Parametreler

size

[in] Yeni tampon boyutu.

SetSymbolPeriod

Sembol ve periyot değerlerini ayarlar.

```
void  SetSymbolPeriod(
   const string           symbol,      // sembol
   const ENUM_TIMEFRAMES  period       // periyot
   )
```

Parametreler

symbol

[in] Yeni sembol.

period

[in] Yeni periyot değeri (ENUM_TIMEFRAMES sayımının değerlerinden biri).

At

Tampon elemanını indisine göre alır.

```
long  At(
   const int  index      // indis
   ) const
```

Parametreler

index

[in] Tampon elemanının indisi.

Dönüş değeri

Belirtilen indisteki tampon elemanı.

Refresh

Tamponu günceller.

```
virtual bool  Refresh()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

RefreshCurrent

Tamponun -sıfır indisli- mevcut değerini günceller.

```
virtual bool  RefreshCurrent()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

CRealVolumeBuffer

CRealVolumeBuffer sınıfı, geçmiş çubukların işlem hacimlerine kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CRealVolumeBuffer sınıfı, geçmiş çubukların işlem hacimlerine kolay erişim sağlar.

Bildirim

```
class CRealVolumeBuffer: public CArrayLong
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayLong
      CRealVolumeBuffer

Sınıf Yöntemleri

Özellikler	
Size	Tampon boyutunu ayarlar
Ayarlar	
SetSymbolPeriod	Sembol ve periyot değerlerini ayarlar
Veri Erişim Yöntemleri	
At	Tampon elemanını indis kullanarak alır
Veri Güncelleme Yöntemleri	
virtual Refresh	Tamponu günceller
virtual RefreshCurrent	Mevcut değeri günceller

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CArrayLong

Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update, Shift, Delete,

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Size

Tampon boyutunu ayarlar.

```
void  Size(
   const int  size      // yeni boyut
   )
```

Parametreler

size

[in] Yeni tampon boyutu.

SetSymbolPeriod

Sembol ve periyot değerlerini ayarlar.

```
void  SetSymbolPeriod(
   const string           symbol,      // sembol
   const ENUM_TIMEFRAMES  period       // periyot
   )
```

Parametreler

symbol

[in] Yeni sembol.

period

[in] Yeni periyot değeri (ENUM_TIMEFRAMES sayımının değerlerinden biri).

At

Tampon elemanını indisine göre alır.

```
long  At(
   const int  index      // indis
   ) const
```

Parametreler

index

[in] Tampon elemanının indisi.

Dönüş değeri

Belirtilen indisteki tampon elemanı.

Refresh

Tamponu günceller.

```
virtual bool  Refresh()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

RefreshCurrent

Tamponun -sıfır indisli- mevcut değerini günceller.

```
virtual bool  RefreshCurrent()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

CDoubleBuffer

CDoubleBuffer sınıfı, double tipli veri tamponlarına kolay erişim için tasarlanmış bir temel sınıftır.

Açıklama

CDoubleBuffer sınıfı, double tipli veri tamponlarına kolay erişim sağlar.

Bildirim

```
class CDoubleBuffer: public CArrayDouble
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayDouble
CDoubleBuffer

İlk nesil

CCloseBuffer, CHighBuffer, CIndicatorBuffer, CLowBuffer, COpenBuffer

Sınıf Yöntemleri

Özellikler	
Size	Tampon boyutunu ayarlar
Ayarlar	
SetSymbolPeriod	Sembol ve periyot değerlerini ayarlar
Veri Erişim Yöntemleri	
At	Tampon elemanını alır
Veri Güncelleme Yöntemleri	
virtual Refresh	Tamponu günceller
virtual RefreshCurrent	Mevcut değeri günceller

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CArrayDouble

Sınıftan türetilen yöntemler 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

Tampon boyutunu ayarlar.

```
void  Size(
   const int  size      // yeni boyut
   )
```

Parametreler

size

[in] Yeni tampon boyutu.

SetSymbolPeriod

Sembol ve periyot değerlerini ayarlar.

```
void  SetSymbolPeriod(
   const string           symbol,      // sembol
   const ENUM_TIMEFRAMES  period       // periyot
   )
```

Parametreler

symbol

[in] Yeni sembol.

period

[in] Yeni periyot değeri (ENUM_TIMEFRAMES sayımının değerlerinden biri).

At

Tampon elemanını indisine göre alır.

```
double  At(
   const int  index      // indis
   ) const
```

Parametreler

index

[in] Tampon elemanının indisi.

Dönüş değeri

Belirtilen indisteki tampon elemanı.

Refresh

Tamponu günceller.

```
virtual bool  Refresh()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

RefreshCurrent

Tamponun -sıfır indisli- mevcut değerini günceller.

```
virtual bool  RefreshCurrent()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

COpenBuffer

COpenBuffer sınıfı, geçmiş çubukların açılış fiyatlarına kolay erişim sağlamak için tasarlanmıştır.

Açıklama

COpenBuffer sınıfı, geçmiş çubukların açılış fiyatlarına erişim sağlar.

Bildirim

```
class COpenBuffer: public CDoubleBuffer
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayDouble
CDoubleBuffer
COpenBuffer

Sınıf Yöntemleri

Veri Güncelleme Yöntemleri	
virtual Refresh	Tamponu günceller
virtual RefreshCurrent	Mevcut değeri günceller

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CDoubleBuffer

Size, At, SetSymbolPeriod

Refresh

Tamponu günceller.

```
virtual bool  Refresh()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

RefreshCurrent

Tamponun -sıfır indisli- mevcut değerini günceller.

```
virtual bool  RefreshCurrent()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

CHighBuffer

CHighBuffer sınıfı, geçmiş çubukların yüksek fiyatlarına kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CHighBuffer sınıfı, geçmiş çubukların yüksek fiyatlarına kolay erişim sağlar.

Bildirim

```
class CHighBuffer: public CDoubleBuffer
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CHighBuffer

Sınıf Yöntemleri

Veri Güncelleme Yöntemleri	
virtual Refresh	Tamponu günceller
virtual RefreshCurrent	Mevcut değeri günceller

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CDoubleBuffer

Size, At, SetSymbolPeriod

Refresh

Tamponu günceller.

```
virtual bool  Refresh()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

RefreshCurrent

Tamponun -sıfır indisli- mevcut değerini günceller.

```
virtual bool  RefreshCurrent()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

CLowBuffer

CLowBuffer sınıfı, geçmiş çubukların düşük fiyatlarına kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CLowBuffer sınıfı, geçmiş çubukların düşük fiyatlarına erişim sağlar.

Bildirim

```
class CLowBuffer: public CDoubleBuffer
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CLowBuffer

Sınıf Yöntemleri

Veri Güncelleme Yöntemleri	
virtual Refresh	Tamponu günceller
virtual RefreshCurrent	Mevcut değeri günceller

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CDoubleBuffer

Size, At, SetSymbolPeriod

Refresh

Tamponu günceller.

```
virtual bool  Refresh()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

RefreshCurrent

Tamponun -sıfır indisli- mevcut değerini günceller.

```
virtual bool  RefreshCurrent()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

CCloseBuffer

CCloseBuffer sınıfı, geçmiş çubukların kapanış fiyatlarına kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CCloseBuffer sınıfı, geçmiş çubukların kapanış fiyatlarına kolay erişim sağlar.

Bildirim

```
class CCloseBuffer: public CDoubleBuffer
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayDouble
CDoubleBuffer
CCloseBuffer

Sınıf Yöntemleri

Veri Güncelleme Yöntemleri	
virtual Refresh	Tamponu günceller
virtual RefreshCurrent	Mevcut değeri günceller

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CDoubleBuffer

Size, At, SetSymbolPeriod

Refresh

Tamponu günceller.

```
virtual bool  Refresh()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

RefreshCurrent

Tamponun -sıfır indisli- mevcut değerini günceller.

```
virtual bool  RefreshCurrent()
```

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

CIndicatorBuffer

CIndicatorBuffer sınıfı, gösterge tamponlarının verilerine kolay erişim için tasarlanmıştır.

Açıklama

CIndicatorBuffer sınıfı, gösterge tamponlarının verilerine kolay erişim için tasarlanmıştır.

Bildirim

```
class CIndicatorBuffer: public CDoubleBuffer
```

Başlık

```
#include <Indicators\Indicator.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayDouble
      CDoubleBuffer
        CIndicatorBuffer

Sınıf Yöntemleri

Özellikler	
Offset	Tampon konumunu alır/ayarlar
Name	Tampon ismini alır/ayarlar
Veri Erişim Yöntemleri	
At	Tampon elemanını alır
Veri Güncelleme Yöntemleri	
Refresh	Tamponu günceller
RefreshCurrent	Sadece mevcut değeri günceller

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CArrayDouble

Delta, Type, Save, Load, Reserve, Resize, Shutdown, Add, AddArray, AddArray, Insert, InsertArray, InsertArray, AssignArray, AssignArray, At, operator, Minimum, Maximum, Update,

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Shift, Delete, DeleteRange, CompareArray, CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast, SearchLinear

Sınıftan türetilen yöntemler CDoubleBuffer

Size, At, SetSymbolPeriod

Offset

Tampon konumunu alır.

```
int  Offset() const
```

Dönüş değeri

Tampon konumu.

Offset

Tampon konumunu ayarlar.

```
void  Offset(
   const int  offset      // konum
   )
```

Parametreler

offset

[in] Yeni tampon konumu.

Name

Tampon ismini alır.

```
string  Name() const
```

Dönüş değeri

Tamponun ismi

Name

Tamponun ismini ayarlar

```
void  Name(
   const string  name      // isim
   )
```

Parametreler

name

[in] Tamponun yeni ismi.

At

Tampon elemanını indisine göre alır.

```
double  At(
   int  index      // indis
   ) const
```

Parametreler

index

[in] Tampon elemanının indisi.

Dönüş değeri

Belirtilen indisteki tampon elemanı.

Refresh

Tüm tamponu günceller.

```
bool  Refresh(
   const int  handle,     // tanıtıcı
   const int  num         // tampon numarası
   )
```

Parametreler

handle

[in] Gösterge tanıtıcı değeri.

num

[in] Gösterge tamponunun indisi.

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

RefreshCurrent

Mevcut (sıfırıncı) tampon elemanını günceller.

```
bool  RefreshCurrent(
   const int  handle,     // gösterge tanıtıcı değeri
   const int  num         // tampon numarası
   )
```

Parametreler

handle

[in] Gösterge tanıtıcı değeri.

num

[in] Tampon numarası.

Dönüş değeri

Başarılı ise 'true', tampon güncellenmediyse 'false'.

CSeries

CSeries, Standart Kütüphanenin zaman-serisi verilerine erişmek için bir temel sınıftır.

Açıklama

CSeries sınıfı, MQL5 zaman-serisi fonksiyonlarına ve onların soyundan gelen tüm yöntemlere kolay erişim sağlar.

Bildirim

```
class CSeries: public CArrayObj
```

Başlık

```
#include <Indicators\Series.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries

İlk nesil

CIndicator, CiRealVolume, CiSpread, CiTickVolume, CiTime, CPriceSeries

Sınıf Yöntemleri

Özellikler	
Name	Zaman-serisinin veya göstergenin ismini alır
BuffersTotal	Zaman-serisinin veya göstergenin tamponlarının sayısını alır
Timeframe	Zaman-serisinin veya göstergenin zaman-dilimleri bayrağını alır
Symbol	Zaman-serisinin veya göstergenin sembolünü alır
Period	Zaman-serisinin veya göstergenin periyodunu alır
RefreshCurrent	Mevcut verinin güncelleme bayrağını alır/ayarlar
Veri Erişim Yöntemleri	
virtual BufferResize	Zaman-serisinin veya göstergenin tampon boyutunu ayarlar
Veri Güncelleme Yöntemleri	
virtual Refresh	Zaman-serisinin veya göstergenin verilerini günceller
PeriodDescription	Periyot değerini dizgi cinsinden alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Zaman-serisinin veya göstergenin ismini alır

```
string  Name() const
```

Dönüş değeri

Zaman-serisinin veya göstergenin ismi.

BuffersTotal

Zaman-serisinin veya göstergenin tamponlarının sayısını alır.

```
int  BuffersTotal() const
```

Dönüş değeri

Zaman-serisinin veya göstergenin tamponlarının sayısını.

Not

Zaman-serileri tek bir tampona sahip olabilir.

Timeframe

Zaman-serisinin veya göstergenin zaman-dilimleri bayrağını alır.

```
int  Timeframe() const
```

Dönüş değeri

Zaman-serisinin veya göstergenin zaman-dilimleri bayrağı.

Not

Bu bayrak, belli zaman-dilimlerinin görünürlüklerini ayarlar.

Symbol

Zaman-serisinin veya göstergenin sembolünü alır.

```
string  Symbol() const
```

Dönüş değeri

Zaman-serisinin veya göstergenin sembolü.

Period

Zaman-serisinin veya göstergenin periyodunu alır.

```
ENUM_TIMEFRAMES  Period() const
```

Dönüş değeri

Zaman-serisinin veya göstergenin periyodu (ENUM_TIMEFRAMES sayımının değerlerinden biri).

RefreshCurrent

Göstergenin veya zaman-serisinin mevcut değerlerini güncellemek için bir bayrak ayarlar.

```
string  RefreshCurrent(
   const bool  flag      // yeni bayrak
   )
```

Parametreler

flag

[in] Yeni bayrak.

Dönüş değeri

Yok.

BufferSize

Gösterge veya zaman-serisi tamponunda mevcut olan veri miktarına dönüş yapar.

```
int  BufferSize() const
```

Dönüş değeri

Gösterge veya zaman-serisi tamponunda mevcut olan veri miktarı.

BufferResize

Zaman-serisinin veya göstergenin tampon boyutunu ayarlar.

```
virtual bool  BufferResize(
   const int  size      // yeni boyut
   )
```

Parametreler

size

[in] Yeni tampon boyutu.

Dönüş değeri

İşlem başarılıysa -true, değilse -false

Not

Tüm gösterge tamponları ve zaman serileri aynı boyuttadır.

Refresh

Zaman-serisinin veya göstergenin verilerini günceller.

```
virtual void  Refresh(
   const int  flags      // bayraklar
   )
```

Parametreler

flags

[in] Güncellenecek zaman-serileri (bayrak).

PeriodDescription

Belirtilen ENUM_TIMEFRAMES sayım değerini dizgi şeklinde alır.

```
string  PeriodDescription(
   const int  val=0      // değer
   )
```

Parametreler

val=0

[in] Dönüştürülecek değer.

Dönüş değeri

Belirtilen ENUM_TIMEFRAMES sayımı değerinin dizgi şekilli ifadesi.

Not

Değer belirtilmemişse veya sıfıra eşitse, zaman-serisinin veya göstergenin zaman-dilimine dönüş yapar.

CPriceSeries

CPriceSeries, fiyat verilerine erişmek için kullanılan temel sınıftır.

Açıklama

CSeries sınıfı, fiyat verileriyle çalışan MQL5 fonksiyonlarına kolay erişim sağlar.

Bildirim

```
class CPriceSeries: public CSeries
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CPriceSeries

İlk nesil

CiClose, CiHigh, CiLow, CiOpen

Sınıf Yöntemleri

Oluşturma Yöntemleri	
virtual BufferResize	Tampon boyutunu ayarlar
Veri Erişim Yöntemleri	
virtual GetData	Belirtilen tampon elemanını indis kullanarak alır
Veri Güncelleme Yöntemleri	
virtual Refresh	Zaman-serisi verilerini günceller
Veri Arama Yöntemleri	
virtual MinIndex	Belirtilen aralıkta yer alan minimal elemanın indisini alır
virtual MinValue	Belirtilen aralıkta yer alan minimal elemanın indisini ve değerini alır
virtual MaxIndex	Belirtilen aralıkta yer alan maksimal elemanın indisini alır
virtual MaxValue	Belirtilen aralıkta yer alan maksimal elemanın indisini ve değerini alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

BufferResize

Tampon için yeni boyut ayarlar.

```
virtual void  BufferResize(
   const int  size      // yeni boyut
   )
```

Parametreler

size

[in] Yeni tampon boyutu.

GetData

Belirtilen tampon elemanını alır.

```
double  GetData(
   const int  index      // indis
   ) const
```

Parametreler

index

[in] Tampon elemanının indisi.

Dönüş değeri

Belirtilen indisteki tampon elemanı veya EMPTY_VALUE.

Refresh

Zaman-serisi verilerini günceller.

```
virtual void  Refresh(
   const int  flags=OBJ_ALL_PERIODS      // zaman-dilimi bayrakları
   )
```

Parametreler

flags=OBJ_ALL_PERIODS

[in] Güncellenecek zaman-serileri (bayrak).

MinIndex

Belirtilen aralıkta yer alan minimal elemanın indisini alır.

```
virtual int  MinIndex(
   const int  start,      // başlangıç indisi
   const int  count       // taranacak elemanların sayısı
   ) const
```

Parametreler

start

[in] Başlangıç indisi.

count

[in] İşlenecek elemanların sayısı.

Dönüş değeri

Belirtilen aralıktaki minimal elemanın indisine, hata durumunda ise -1 değerine dönüş yapar.

MinValue

Belirtilen aralıkta yer alan minimal elemanın indisini ve değerini alır.

```
virtual double  MinValue(
   const int    start,      // başlangıç indisi
   const int    count,      // taranacak eleman sayısı
   int&         index       // indis değişkeni için referans
   ) const
```

Parametreler

start

[in] Başlangıç indisi.

count

[in] İşlenecek elemanların sayısı.

index

[out] Tamsayı tipli referans değişkeni.

Dönüş değeri

Belirtilen tamponun belirtilen aralığındaki minimal elemanın değerine, hata durumunda ise EMPTY_VALUE değerine dönüş yapar.

Not

Minimal elemanın indisi referansla geçirilen değişkenin indisine kaydedilir.

MaxIndex

Belirtilen aralıkta yer alan maksimal elemanın indisini alır

```
virtual int  MaxIndex(
   const int  start,      // başlangıç indisi
   const int  count       // taranacak elemanların sayısı
   ) const
```

Parametreler

start

[in] Başlangıç indisi.

count

[in] İşlenecek elemanların sayısı.

Dönüş değeri

Belirtilen tamponun belirtilen aralığındaki maksimal elemanın indisine, hata durumunda ise -1 değerine dönüş yapar.

MaxValue

Belirtilen aralıkta yer alan maksimal elemanın indisini ve değerini alır.

```
virtual double  MaxValue(
   const int   start,     // başlangıç indisi
   const int   count,     // taranacak eleman sayısı
   int&        index      // indis değişkeni için referans
   ) const
```

Parametreler

start

[in] Başlangıç indisi.

count

[in] İşlenecek elemanların sayısı.

index

[out] Tamsayı tipli referans değişkeni.

Dönüş değeri

Belirtilen tamponun belirtilen aralığındaki maksimal elemanın değerine, hata durumunda ise EMPTY_VALUE değerine.

Not

Maksimal elemanın indisi referansla geçirilen değişkenin indisine kaydedilir.

CIndicator

CIndicator sınıfı, MQL5 Standart Kütüphanesi içinde yer alan teknik göstergeler için bir temel sınıftır.

Açıklama

CIndicator sınıfı, soyundan gelen tüm sınıflar için MQL5 API teknik gösterge fonksiyonlarına kolay erişim sağlar.

Bildirim

```
class CIndicator: public CSeries
```

Başlık

```
#include <Indicators\Indicator.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator

İlk nesil

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

Sınıf Yöntemleri

Özellikler	
Handle	Göstergenin tanıtıcısını alır.
Status	Gösterge durumunu alır.
FullRelease	Gösterge tanıtıcısı için serbest bırakma bayrağı ayarlar.
Oluşturma	
Create	Gösterge oluşturur
BufferResize	Yeni tampon boyutu ayarlar
Veri Erişim Yöntemleri	
GetData	Gösterge tamponundan veri kopyalar
Veri Güncelleme Yöntemleri	
Refresh	Gösterge verisini günceller

Özellikler	
Min/Max değerlerini bulur	
Minimum	Belirtilen tamponun belirtilen aralığındaki minimal elemanın indisini alır.
MinValue	Belirtilen tamponun belirtilen aralığındaki minimal elemanın indisini ve değerini alır.
Maximum	Belirtilen tamponun belirtilen aralığındaki maksimal elemanın indisini alır.
MaxValue	Belirtilen tamponun belirtilen aralığındaki maksimal elemanın indisini ve değerini alır.
Sayım Dönüşümü	
MethodDescription	ENUM_MA_METHOD sayımının değerini dizgi şeklinde alır
PriceDescription	ENUM_APPLIED_PRICE sayımının değerini dizgi şeklinde alır
VolumeDescription	ENUM_APPLIED_VOLUME sayımının değerini dizgi şeklinde alır
Çizelge Çalışmaları	
AddToChart	Göstergeyi çizelgeye ekler
DeleteFromChart	Göstergeyi çizelgeden siler

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Türetilmiş sınıflar:

- CiAC
- CiAD
- CiADX
- CiADXWilder
- CiAlligator
- CiAMA

- CiAO
- CiATR
- CiBands
- CiBearsPower
- CiBullsPower
- CiBWMFI
- CiCCI
- CiChaikin
- CiDEMA
- CiDeMarker
- CiEnvelopes
- CiForce
- CiFractals
- CiFrAMA
- CiGator
- CiIchimoku
- CiMA
- CiMACD
- CiMFI
- CiMomentum
- CiOBV
- CiOsMA
- CiRSI
- CiRVI
- CiSAR
- CiStdDev
- CiStochastic
- CiTEMA
- CiTriX
- CiVIDyA
- CiVolumes
- CiWPR

Handle

Göstergenin tanıtıcısını alır.

```
int  Handle() const
```

Dönüş değeri

Handle of the indicator.

Status

Gösterge durumunu alır.

```
string  Status() const
```

Dönüş değeri

Gösterge oluşturma durumu.

FullRelease

Gösterge tanıtıcısı için serbest bırakma bayrağı ayarlar.

```
void  FullRelease(
   const bool  flag=true      // bayrak
   )
```

Parametreler

flag

[in] Tanıtıcı serbest bırakma bayrağının yeni değeri.

Create

Belirtilen parametrelerle gösterge oluşturur.

```
bool  Create(
   const string           symbol,          // sembol
   const ENUM_TIMEFRAMES  period,          // periyot
   const ENUM_INDICATOR   type,            // tip
   const int              num_params,      // parametrelerin sayısı
   const MqlParam&        params[]         // parametre dizisinin referansı
   )
```

Parametreler

symbol

[in] Sembol ismi.

period

[in] Periyot (ENUM_TIMEFRAMES sayımının değerlerinden biri).

type

[in] Gösterge tipi (ENUM_INDICATOR sayımının değerlerinden biri).

num_params

[in] Gösterge parametrelerinin sayısı.

params

[in] Gösterge için parametre dizisinin referansı.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

BufferResize

Gösterge tamponunun boyutunu ayarlar ayarlar

```
virtual bool  BufferResize(
   const int  size      // boyut
   )
```

Parametreler

size

[in] Yeni tampon boyutu.

Dönüş değeri

İşlem başarılıysa -true, değilse -false

Not

Tüm gösterge tamponları aynı boyuttadır.

BarsCalculated

Gösterge için hesaplanmış çubuk sayısına dönüş yapar.

```
int  BarsCalculated() const;
```

Dönüş Değeri

Gösterge tamponunda hesaplanmış veri miktarına veya hata durumunda (henüz herhangi bir veri hesaplanmamışsa) -1 değerine dönüş yapar.

GetData

Gösterge tamponunun belirtilen konumundaki elemanı alır. Yöntem kullanılmadan önce güncel verilerle çalışmak için Refresh() çağrısı yapılmalıdır.

```
double  GetData(
   const int  buffer_num,     // tampon numarası
   const int  index           // eleman indisi
   ) const
```

Parametreler

buffer_num

[in] Tampon numarası.

index

[in] Elemanın indisi.

Dönüş değeri

Başarılı olması durumunda elemanın sayısal değerine, hata durumunda ise EMPTY_VALUE değerine dönüş yapar.

GetData

Gösterge tamponundan belirtilen başlangıç konumuna ve istenilen veri miktarına göre veri alır.

```
int  GetData(
   const int      start_pos,      // pozisyon
   const int      count,          // eleman sayısı
   const int      buffer_num,     // tampon numarası
   double&        buffer[]        // veri için hedef dizi
   ) const
```

Parametreler

start_pos

[in] Gösterge tamponunun başlangıç konumu.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer_num

[in] Gösterge tamponunun numarası.

buffer

[in] Hedef veri dizisinin referansı.

Dönüş değeri

Başarı durumunda belirtilen tampondan alınan eleman sayısına, aksi durumda -1 değerine dönüş yapar.

GetData

Gösterge tamponundan belirtilen başlangıç zamanına ve istenilen veri miktarına göre veri alır.

```
int  GetData(
   const datetime  start_time,     // başlangıç zamanı
   const int       count,          // istenen eleman sayısı
   const int       buffer_num,     // tampon numarası
   double&         buffer[]        // veri için hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer_num

[in] Gösterge tamponunun numarası.

buffer

[in] Hedef dizinin referansı.

Dönüş değeri

Başarı durumunda belirtilen tampondan alınan eleman sayısına, aksi durumda -1 değerine dönüş yapar.

GetData

Gösterge tamponundan belirtilen başlangıç ve bitiş zamanına göre veri alır.

```
int  GetData(
   const datetime  start_time,     // başlangıç zamanı
   const datetime  stop_time,      // bitiş zamanı
   const int       buffer_num,     // tampon numarası
   double&         buffer[]        // veri için hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

stop_time

[in] Bitiş zamanı.

buffer_num

[in] Gösterge tamponunun numarası.

buffer

[in] Hedef dizinin referansı.

Dönüş değeri

Başarı durumunda belirtilen tampondan alınan eleman sayısına, aksi durumda -1 değerine dönüş yapar.

Refresh

Gösterge verisini günceller. GetData() yönteminden önce çağrılması önerilir.

```
virtual void  Refresh(
   int  flags=OBJ_ALL_PERIODS      // bayraklar
   )
```

Parametreler

flags=OBJ_ALL_PERIODS

[in] Zaman-dilimi güncelleme bayrağı.

Minimum

Belirtilen tamponun belirtilen aralığındaki minimal elemanın indisini alır.

```
int  Minimum(
   const int  buffer_num,      // tampon numarası
   const int  start,           // başlangıç indisi
   const int  count            // işlenecek elemnların sayısı
   ) const
```

Parametreler

buffer_num

[in] Elemanların aranacağı tamponun numarası.

start

[in] Arama için başlangıç indisi.

count

[in] Aranacak elemanların sayısı.

Dönüş değeri

Belirtilen tamponun belirtilen aralığındaki minimal elemanın indisi.

MinValue

Belirtilen tamponun belirtilen aralığındaki minimal elemanın indisini ve değerini alır.

```
double  MinValue(
   const int   buffer_num,     // tampon numarası
   const int   start,          // başlangıç indisi
   const int   count,          // işlenecek elemanların sayısı
   int&        index           // referans
   ) const
```

Parametreler

buffer_num

[in] Elemanların aranacağı tamponun numarası.

start

[in] Başlangıç indisi.

count

[in] İşlenecek elemanların sayısı.

index

[out] Minimal elemanın indisi için int tipli referans değişkeni.

Dönüş değeri

Belirtilen tamponun belirtilen aralığındaki minimal elemanın değeri.

Not

Minimal elemanın indisi referansla geçirilen değişkenin indisine kaydedilir.

Maximum

Belirtilen tamponun belirtilen aralığındaki maksimal elemanın indisini alır.

```
int  Maximum(
   const int  buffer_num,      // tampon numarası
   const int  start,           // başlangıç indisi
   const int  count            // işlenecek elemnların sayısı
   ) const
```

Parametreler

buffer_num

[in] Elemanların aranacağı tamponun numarası.

start

[in] Arama için başlangıç indisi.

count

[in] Aranacak elemanların sayısı.

Dönüş değeri

Belirtilen tamponun belirtilen aralığındaki maksimal elemanın indisi.

MaxValue

Belirtilen tamponun belirtilen aralığındaki maksimal elemanın indisini ve değerini alır.

```
double  MaxValue(
   const int   buffer_num,     // tampon numarası
   const int   start,          // başlangıç indisi
   const int   count,          // işlenecek elemanların sayısı
   int&        index           // referans
   ) const
```

Parametreler

buffer_num

[in] Elemanların aranacağı tamponun numarası.

start

[in] Başlangıç indisi.

count

[in] İşlenecek elemanların sayısı.

index

[out] Maksimal elemanın indisi için int tipli referans değişkeni.

Dönüş değeri

Belirtilen tamponun belirtilen aralığındaki maksimal elemanın indisi.

Not

Maksimal elemanın indisi referansla geçirilen değişkenin indisine kaydedilir.

MethodDescription

Bu fonksiyon ENUM_MA_METHOD sayımının değerini dizgi şeklinde alır.

```
string  MethodDescription(
   const int  val      // değer
   ) const
```

Parametreler

val

[in] ENUM_MA_METHOD sayımının değeri

Dönüş değeri

ENUM_MA_METHOD sayımının dizgi şeklindeki değeri.

PriceDescription

Bu yöntem ENUM_APPLIED_PRICE sayımının değerini dizgi şeklindealır.

```
string  PriceDescription(
   const int  val      // değer
   ) const
```

Parametreler

val

[in] ENUM_APPLIED_PRICE sayımının değeri.

Dönüş değeri

ENUM_APPLIED_PRICEsayımının dizgi şeklinde değeri.

VolumeDescription

Bu yöntem, ENUM_APPLIED_VOLUME sayımının değerine dizgi şeklinde dönüş yapar.

```
string  VolumeDescription(
   const int  val      // değer
   ) const
```

Parametreler

val

[in] ENUM_APPLIED_VOLUME sayımının değeri.

Dönüş değeri

ENUM_APPLIED_VOLUME sayımının dizgi şeklindeki değeri.

AddToChart

Göstergeyi çizelgeye ekler.

```
bool  AddToChart(
   const long  chart,      // çizelge tanımlayıcısı
   const int  subwin       // çizelge alt-pencresi
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

subwin

[in] Çizelge alt-penceresi.

Dönüş değeri

Başarılı ise 'true', hata durumunda 'false'.

DeleteFromChart

Göstergeyi çizelgeden siler.

```
bool  DeleteFromChart(
   const long  chart,      // çizelge tanımlayıcısı
   const int   subwin      // çizelge alt-pencresi
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

subwin

[in] Çizelge alt-penceresi.

Dönüş değeri

Başarılı ise 'true', hata durumunda 'false'.

CIndicators

CIndicators, teknik gösterge ve zaman-serisi sınıflarının örneklerini toplamak için tasarlanmış bir sınıftır.

Açıklama

CIndicators sınıfı, teknik gösterge sınıflarının örneklerinin oluşturulmasını, saklanmasını ve yönetilmesini (veri eşlemesi, işleme ve bellek yönetimi) sağlar.

Bildirim

```
class CIndicators: public CArrayObj
```

Başlık

```
#include <Indicators\Indicators.mqh>
```

Sınıf Yöntemleri

Oluşturma Yöntemleri	
Create	Teknik gösterge oluşturur
Veri Güncelleme Yöntemleri	
Refresh	Koleksiyondaki tüm teknik göstergelerin verilerini günceller

Create

Belirtilen parametrelerle bir gösterge oluşturur.

```
CIndicator*  Create(
   const string           symbol,     // sembol ismi
   const ENUM_TIMEFRAMES  period,     // periyot
   const ENUM_INDICATOR   type,       // gösterge tipi
   const int              count,      // parametrelerin sayısı
   const MqlParam&        params      // parametre dizisinin referansı
   )
```

Parametreler

symbol

[in] Sembol ismi.

period

[in] Periyot (ENUM_TIMEFRAMES sayımının değerlerinden biri).

type

[in] Gösterge tipi (ENUM_INDICATOR sayımının değerlerinden biri).

count

[in] Gösterge parametrelerinin sayısı.

params

[in] Gösterge için parametre dizisinin referansı.

Dönüş değeri

Başarılı ise oluşturulan göstergenin referansına, gösterge oluşturulamadıysa 'NULL' değerine dönüş yapar.

Refresh

Koleksiyon içindeki tüm göstergelerin verilerini günceller.

```
int  Refresh()
```

Dönüş değeri

Güncellenmiş zaman-dilimi bayraklarına (nesne görünürlük bayrağı şeklinde) dönüş yapar.

Zaman-serisi sınıfları

Bu bölüm, zaman-serileri sınıflarıyla çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için açıklamalar içermektedir.

Sınıf	Açıklama
CiSpread	Spread (alış satış farkı) serisinin geçmiş verilerine erişim sağlar
CiTime	Geçmiş çubukların açılış zamanı verilerine erişim sağlar
CiTickVolume	Geçmiş çubukların tik hacmi verilerine erişim sağlar
CiRealVolume	Geçmiş çubukların işlem hacmi verilerine erişim sağlar
CiOpen	Geçmiş çubukların açılış fiyatı verilerine erişim sağlar
CiHigh	Geçmiş çubukların yüksek fiyat verilerine erişim sağlar
CiLow	Geçmiş çubukların düşük fiyat verilerine erişim sağlar
CiClose	Geçmiş çubukların kapanış fiyatı verilerine erişim sağlar

CiSpread

CiSpread sınıfı, geçmiş çubukların spread (alış satış farkı) değerlerine kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CiSpread sınıfı, spread serisinin geçmiş verilerine erişim sağlar.

Bildirim

```
class CiSpread: public CSeries
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CiSpread

Sınıf Yöntemleri

Oluşturma Yöntemleri	
Create	Bir zaman-serisi oluşturur
BufferResize	Tampon boyutunu ayarlar
Veri Erişim Yöntemleri	
GetData	Veriyi alır
Veri Güncelleme Yöntemleri	
Refresh	Verileri günceller

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Create

Geçmiş çubukların spread verilerine kolay erişim sağlamak için belirtilen parametrelerle bir zaman-serisi oluşturur.

```
bool  Create(
   string           symbol,      // sembol
   ENUM_TIMEFRAMES  period       // periyot
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerleri).

Dönüş değeri

Başarılı ise 'true', zaman-serisi oluşturulamadıysa 'false'.

BufferResize

Zaman-serisi için yeni boyut ayarlar.

```
virtual void  BufferResize(
   int  size      // yeni değer
   )
```

Parametreler

size

[in] Yeni tampon boyutu.

GetData

Zaman-serisinin elemanını indis kullanarak alır.

```
int  GetData(
   int  index      // indis
   ) const
```

Parametreler

index

[in] İstenen elemanın indisi.

Dönüş değeri

Zaman-serisi elemanı veya 0.

GetData

Belirtilen başlangıç konumuna ve istenilen veri miktarına göre zaman-serisinden veri alır.

```
int  GetData(
   int   start_pos,     // başlangıç konumu
   int   count,         // alınacak elemanların sayısı
   int&  buffer         // hedef dizi
   ) const
```

Parametreler

start_pos

[in] Zaman-serisinin başlangıç konumu.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer

[in] Hedef veri dizisinin referansı.

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

GetData

Belirtilen başlangıç zamanına ve istenilen veri miktarına göre zaman-serisinden veri alır.

```
int  GetData(
   datetime  start_time,     // başlangıç zamanı
   int       count,          // eleman sayısı
   int&      buffer          // hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer

[in] Hedef veri dizisinin referansı.

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

GetData

Belirtilen başlangıç ve bitiş zamanlarına göre zaman-serisinden veri alır.

```
int  GetData(
   datetime  start_time,     // başlangıç zamanı
   datetime  stop_time,      // bitiş zamanı
   int&      buffer          // hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

stop_time

[in] Bitiş zamanı.

buffer

[in] Hedef veri dizisinin referansı

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

Refresh

Zaman-serisi verilerini günceller.

```
virtual void  Refresh(
   int  flags      // bayraklar
   )
```

Parametreler

flags

[in] Zaman-serisi bayrakları.

CiTime

CiTime sınıfı, geçmiş çubukların açılış zamanlarına kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CiTime sınıfı, geçmiş çubukların açılış zamanlarına erişim sağlar.

Bildirim

```
class CiTime: public CSeries
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CiTime

Sınıf Yöntemleri

Oluşturma Yöntemleri	
Create	Bir zaman-serisi oluşturur
BufferResize	Tampon boyutunu ayarlar
Veri Erişim Yöntemleri	
GetData	Veriyi alır
Veri Güncelleme Yöntemleri	
Refresh	Verileri günceller

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Create

Geçmiş çubukların açılış zamanlarına kolay erişim sağlamak için belirtilen parametrelerle bir zaman-serisi oluşturur.

```
bool  Create(
   string           symbol,      // sembol
   ENUM_TIMEFRAMES  period       // periyot
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', zaman-serisi oluşturulamadıysa 'false'.

BufferResize

Zaman-serisi için yeni boyut ayarlar.

```
virtual void  BufferResize(
   int  size      // yeni değer
   )
```

Parametreler

size

[in] Yeni tampon boyutu.

GetData

Zaman-serisinin elemanını indis kullanarak alır.

```
datetime  GetData(
   int  index      // indis
   ) const
```

Parametreler

index

[in] İstenen elemanın indisi.

Dönüş değeri

Zaman-serisi elemanı veya 0.

GetData

Belirtilen başlangıç konumuna ve istenilen veri miktarına göre zaman-serisinden veri alır.

```
int  GetData(
   int   start_pos,      // başlangıç konumu
   int   count,          // alınacak elemanların sayısı
   long& buffer          // hedef dizi
   ) const
```

Parametreler

start_pos

[in] Zaman-serisinin başlangıç konumu.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer

[in] Hedef veri dizisinin referansı.

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

GetData

Belirtilen başlangıç zamanına ve istenilen veri miktarına göre zaman-serisinden veri alır.

```
int  GetData(
   datetime  start_time,     // başlangıç zamanı
   int       count,          // eleman sayısı
   long&     buffer          // hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer

[in] Hedef veri dizisinin referansı.

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

GetData

Belirtilen başlangıç ve bitiş zamanlarına göre zaman-serisinden veri alır.

```
int  GetData(
   datetime  start_time,     // başlangıç zamanı
   datetime  stop_time,      // bitiş zamanı
   long&     buffer          // hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

stop_time

[in] Bitiş zamanı.

buffer

[in] Hedef veri dizisinin referansı

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

Refresh

Zaman-serisi verilerini günceller.

```
virtual void  Refresh(
   int  flags      // bayraklar
   )
```

Parametreler

flags

[in] Zaman-serisi bayrakları.

CiTickVolume

CiTickVolume sınıfı, geçmiş çubukların tik hacimlerine kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CiTickVolume sınıfı, geçmiş çubukların tik hacimlerine kolay sağlar.

Bildirim

```
class CiTickVolume: public CSeries
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CiTickVolume

Sınıf Yöntemleri

Oluşturma Yöntemleri	
Create	Bir zaman-serisi oluşturur
BufferResize	Tampon boyutunu ayarlar
Veri Erişim Yöntemleri	
GetData	Veriyi alır
Veri Güncelleme Yöntemleri	
Refresh	Verileri günceller

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Create

Geçmiş çubukların tik hacimlerine kolay erişim sağlamak için belirtilen parametrelerle bir zaman-serisi oluşturur.

```
bool  Create(
   string           symbol,      // sembol
   ENUM_TIMEFRAMES  period       // periyot
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', zaman-serisi oluşturulamadıysa 'false'.

BufferResize

Zaman-serisi için yeni boyut ayarlar.

```
virtual void  BufferResize(
   int  size      // yeni değer
   )
```

Parametreler

size

[in] Yeni tampon boyutu.

GetData

Zaman-serisinin elemanını indis kullanarak alır.

```
long  GetData(
   int  index      // indis
   ) const
```

Parametreler

index

[in] İstenen elemanın indisi.

Dönüş değeri

Zaman-serisi elemanı veya 0.

GetData

Belirtilen başlangıç konumuna ve istenilen veri miktarına göre zaman-serisinden veri alır.

```
int  GetData(
   int   start_pos,      // başlangıç konumu
   int   count,          // alınacak elemanların sayısı
   long& buffer          // hedef dizi
   ) const
```

Parametreler

start_pos

[in] Zaman-serisinin başlangıç konumu.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer

[in] Hedef veri dizisinin referansı.

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

GetData

Belirtilen başlangıç zamanına ve istenilen veri miktarına göre zaman-serisinden veri alır.

```
int  GetData(
   datetime  start_time,     // başlangıç zamanı
   int       count,          // eleman sayısı
   long&     buffer          // hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer

[in] Hedef veri dizisinin referansı.

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

GetData

Belirtilen başlangıç ve bitiş zamanlarına göre zaman-serisinden veri alır.

```
int  GetData(
   datetime  start_time,     // başlangıç zamanı
   datetime  stop_time,      // bitiş zamanı
   long&     buffer          // hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

stop_time

[in] Bitiş zamanı.

buffer

[in] Hedef veri dizisinin referansı

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

Refresh

Zaman-serisi verilerini günceller.

```
virtual void  Refresh(
   int  flags      // bayraklar
   )
```

Parametreler

flags

[in] Zaman-serisi bayrakları.

CiRealVolume

CiRealVolume sınıfı, geçmiş çubukların işlem hacimlerine kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CiRealVolume sınıfı, geçmiş çubukların işlem hacimlerine kolay erişim sağlar.

Bildirim

```
class CiRealVolume: public CSeries
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CiRealVolume

Sınıf Yöntemleri

Oluşturma Yöntemleri	
Create	Bir zaman-serisi oluşturur
BufferResize	Tampon boyutunu ayarlar
Veri Erişim Yöntemleri	
GetData	Veriyi alır
Veri Güncelleme Yöntemleri	
Refresh	Verileri günceller

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Create

Geçmiş çubukların işlem hacimlerine kolay erişim sağlamak için belirtilen parametrelerle bir zaman-serisi oluşturur.

```
bool  Create(
   string           symbol,      // sembol
   ENUM_TIMEFRAMES  period       // periyot
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', zaman-serisi oluşturulamadıysa 'false'.

BufferResize

Zaman-serisi için yeni boyut ayarlar.

```
virtual void  BufferResize(
   int  size      // yeni değer
   )
```

Parametreler

size

[in] Yeni tampon boyutu.

GetData

Zaman-serisinin elemanını indis kullanarak alır.

```
datetime  GetData(
   int  index      // indis
   ) const
```

Parametreler

index

[in] İstenen elemanın indisi.

Dönüş değeri

Zaman-serisi elemanı veya 0.

GetData

Belirtilen başlangıç konumuna ve istenilen veri miktarına göre zaman-serisinden veri alır.

```
int  GetData(
   int   start_pos,      // başlangıç konumu
   int   count,          // alınacak elemanların sayısı
   long& buffer          // hedef dizi
   ) const
```

Parametreler

start_pos

[in] Zaman-serisinin başlangıç konumu.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer

[in] Hedef veri dizisinin referansı.

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

GetData

Belirtilen başlangıç zamanına ve istenilen veri miktarına göre zaman-serisinden veri alır.

```
int  GetData(
   datetime  start_time,     // başlangıç zamanı
   int       count,          // eleman sayısı
   long&     buffer          // hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer

[in] Hedef veri dizisinin referansı.

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

GetData

Belirtilen başlangıç ve bitiş zamanlarına göre zaman-serisinden veri alır.

```
int  GetData(
   datetime  start_time,     // başlangıç zamanı
   datetime  stop_time,      // bitiş zamanı
   long&     buffer          // hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

stop_time

[in] Bitiş zamanı.

buffer

[in] Hedef veri dizisinin referansı

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

Refresh

Zaman-serisi verilerini günceller.

```
virtual void  Refresh(
   int  flags      // bayraklar
   )
```

Parametreler

flags

[in] Zaman-serisi bayrakları.

CiOpen

CiOpen sınıfı, geçmiş çubukların açılış fiyatlarına kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CiOpen sınıfı, geçmiş çubukların açılış fiyatlarına erişim sağlar.

Bildirim

```
class CiOpen: public CPriceSeries
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiOpen

Sınıf Yöntemleri

Oluşturma Yöntemleri	
Create	Bir zaman-serisi oluşturur
Veri Erişim Yöntemleri	
GetData	Veriyi alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

Create

Geçmiş çubukların açılış fiyatlarına kolay erişim sağlamak için belirtilen parametrelerle bir zaman-serisi oluşturur.

```
bool  Create(
   string           symbol,      // sembol
   ENUM_TIMEFRAMES  period       // periyot
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', zaman-serisi oluşturulamadıysa 'false'.

GetData

Belirtilen başlangıç konumuna ve istenilen veri miktarına göre zaman-serisinden veri alır.

```
int  GetData(
   int    start_pos,      // başlangıç konumu
   int    count,          // alınacak elemanların sayısı
   double& buffer         // hedef dizi
   ) const
```

Parametreler

start_pos

[in] Zaman-serisinin başlangıç konumu.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer

[in] Hedef veri dizisinin referansı.

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

GetData

Belirtilen başlangıç zamanına ve istenilen veri miktarına göre zaman-serisinden veri alır.

```
int  GetData(
   datetime  start_time,      // başlangıç zamanı
   int       count,           // eleman sayısı
   double&     buffer         // hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer

[in] Hedef veri dizisinin referansı.

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

GetData

Belirtilen başlangıç ve bitiş zamanlarına göre zaman-serisinden veri alır.

```
int  GetData(
   datetime  start_time,      // başlangıç zamanı
   datetime  stop_time,       // bitiş zamanı
   double&   buffer           // hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

stop_time

[in] Bitiş zamanı.

buffer

[in] Hedef veri dizisinin referansı

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

CiHigh

CHigh sınıfı, geçmiş çubukların yüksek fiyatlarına kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CiHigh sınıfı, geçmiş çubukların yüksek fiyatlarına kolay erişim sağlar.

Bildirim

```
class CiHigh: public CPriceSeries
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiHigh

Sınıf Yöntemleri

Oluşturma Yöntemleri	
Create	Bir zaman-serisi oluşturur
Veri Erişim Yöntemleri	
GetData	Veriyi alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

Create

Geçmiş çubukların yüksek fiyatlarına kolay erişim sağlamak için belirtilen parametrelerle bir zaman-serisi oluşturur.

```
bool  Create(
   string           symbol,      // sembol
   ENUM_TIMEFRAMES  period       // periyot
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', zaman-serisi oluşturulamadıysa 'false'.

GetData

Belirtilen başlangıç konumuna ve istenilen veri miktarına göre zaman-serisinden veri alır.

```
int  GetData(
   int    start_pos,      // başlangıç konumu
   int    count,          // alınacak elemanların sayısı
   double& buffer         // hedef dizi
   ) const
```

Parametreler

start_pos

[in] Zaman-serisinin başlangıç konumu.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer

[in] Hedef veri dizisinin referansı.

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

GetData

Belirtilen başlangıç zamanına ve istenilen veri miktarına göre zaman-serisinden veri alır.

```
int  GetData(
   datetime  start_time,      // başlangıç zamanı
   int       count,           // eleman sayısı
   double&     buffer         // hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer

[in] Hedef veri dizisinin referansı.

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

GetData

Belirtilen başlangıç ve bitiş zamanlarına göre zaman-serisinden veri alır.

```
int  GetData(
   datetime  start_time,      // başlangıç zamanı
   datetime  stop_time,       // bitiş zamanı
   double&   buffer           // hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

stop_time

[in] Bitiş zamanı.

buffer

[in] Hedef veri dizisinin referansı

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

CiLow

CiLow sınıfı, geçmiş çubukların düşük fiyatlarına kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CiLow sınıfı, geçmiş çubukların düşük fiyatlarına erişim sağlar.

Bildirim

```
class CiLow: public CPriceSeries
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CPriceSeries
          CiLow

Sınıf Yöntemleri

Oluşturma Yöntemleri	
Create	Bir zaman-serisi oluşturur
Veri Erişim Yöntemleri	
GetData	Veriyi alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

Create

Geçmiş çubukların düşük fiyatlarına kolay erişim sağlamak için belirtilen parametrelerle bir zaman-serisi oluşturur.

```
bool  Create(
   string           symbol,      // sembol
   ENUM_TIMEFRAMES  period       // periyot
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', zaman-serisi oluşturulamadıysa 'false'.

GetData

Belirtilen başlangıç konumuna ve istenilen veri miktarına göre zaman-serisinden veri alır.

```
int  GetData(
   int    start_pos,      // başlangıç konumu
   int    count,          // alınacak elemanların sayısı
   double& buffer         // hedef dizi
   ) const
```

Parametreler

start_pos

[in] Zaman-serisinin başlangıç konumu.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer

[in] Hedef veri dizisinin referansı.

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

GetData

Belirtilen başlangıç zamanına ve istenilen veri miktarına göre zaman-serisinden veri alır.

```
int  GetData(
   datetime  start_time,      // başlangıç zamanı
   int       count,           // eleman sayısı
   double&     buffer         // hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer

[in] Hedef veri dizisinin referansı.

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

GetData

Belirtilen başlangıç ve bitiş zamanlarına göre zaman-serisinden veri alır.

```
int  GetData(
   datetime  start_time,      // başlangıç zamanı
   datetime  stop_time,       // bitiş zamanı
   double&   buffer           // hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

stop_time

[in] Bitiş zamanı.

buffer

[in] Hedef veri dizisinin referansı

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

CiClose

CiClose sınıfı, geçmiş çubukların kapanış fiyatlarına kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CiClose sınıfı, geçmiş çubukların kapanış fiyatlarına kolay erişim sağlar.

Bildirim

```
class CiClose: public CPriceSeries
```

Başlık

```
#include <Indicators\TimeSeries.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CPriceSeries
CiClose

Sınıf Yöntemleri

Oluşturma Yöntemleri	
Create	Bir zaman-serisi oluşturur
Veri Erişim Yöntemleri	
GetData	Veriyi alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CPriceSeries

BufferResize, MinIndex, MinValue, MaxIndex, MaxValue, GetData, Refresh

Create

Geçmiş çubukların kapanış fiyatlarına kolay erişim sağlamak için belirtilen parametrelerle bir zaman-serisi oluşturur.

```
bool  Create(
   string            symbol,      // sembol
   ENUM_TIMEFRAMES   period       // periyot
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', zaman-serisi oluşturulamadıysa 'false'.

GetData

Belirtilen başlangıç konumuna ve istenilen veri miktarına göre zaman-serisinden veri alır.

```
int  GetData(
   int    start_pos,      // başlangıç konumu
   int    count,          // alınacak elemanların sayısı
   double& buffer         // hedef dizi
   ) const
```

Parametreler

start_pos

[in] Zaman-serisinin başlangıç konumu.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer

[in] Hedef veri dizisinin referansı.

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

GetData

Belirtilen başlangıç zamanına ve istenilen veri miktarına göre zaman-serisinden veri alır.

```
int  GetData(
   datetime  start_time,      // başlangıç zamanı
   int       count,           // eleman sayısı
   double&     buffer         // hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

count

[in] İhtiyaç duyulan eleman sayısı.

buffer

[in] Hedef veri dizisinin referansı.

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

GetData

Belirtilen başlangıç ve bitiş zamanlarına göre zaman-serisinden veri alır.

```
int  GetData(
   datetime  start_time,      // başlangıç zamanı
   datetime  stop_time,       // bitiş zamanı
   double&   buffer           // hedef dizi
   ) const
```

Parametreler

start_time

[in] Başlangıç zamanı.

stop_time

[in] Bitiş zamanı.

buffer

[in] Hedef veri dizisinin referansı

Dönüş değeri

Başarılıysa >=0, hata durumunda -1 değerlerine dönüş yapar.

Trend Göstergesi sınıfları

Bu bölüm, Trend göstergelerinin sınıflarıyla çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için açıklamalar içermektedir.

Sınıf/grup	Açıklama
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 sınıfı Average Directional Index teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiADX sınıfı, Average Directional Index göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiADX: public CIndicator
```

Başlık

```
#include <Indicators\Trend.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiADX

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Ana çizgi tamponunun elemanına dönüş yapar
Plus	+DI çizgisi tamponunun elemanına dönüş yapar
Minus	-DI çizgisi tamponunun elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string            symbol,         // sembol
   ENUM_TIMEFRAMES   period,         // periyot
   int               ma_period       // ortalama periyodu
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Plus

+DI çizgisinin tamponu üzerinde, belirtilen indisteki elemana dönüş yapar.

```
double   Plus(
   int   index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

+DI çizgisinin tamponu üzerinde, belirtilen indisteki elemana veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

Minus

-DI çizgisinin tamponu üzerinde, belirtilen indisteki elemana dönüş yapar.

```
double  Minus(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

-DI çizgisinin tamponu üzerinde, belirtilen indisteki elemana veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiADX için IND_ADX).

CiADXWilder

CiADXWilder sınıfı Average Directional Index by Welles Wilder teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiADXWilder sınıfı, Average Directional Index by Welles Wilder göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir..

Bildirim

```
class CiADXWilder: public CIndicator
```

Başlık

```
#include <Indicators\Trend.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiADXWilder

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Ana çizgi tamponunun elemanına dönüş yapar
Plus	+DI çizgisi tamponunun elemanına dönüş yapar
Minus	-DI çizgisi tamponunun elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string            symbol,         // sembol
   ENUM_TIMEFRAMES   period,         // periyot
   int               ma_period       // ortalama periyodu
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Plus

+DI çizgisinin tamponu üzerinde, belirtilen indisteki elemana dönüş yapar.

```
double   Plus(
   int   index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

+DI çizgisinin tamponu üzerinde, belirtilen indisteki elemana veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

Minus

-DI çizgisinin tamponu üzerinde, belirtilen indisteki elemana dönüş yapar.

```
double  Minus(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

-DI çizgisinin tamponu üzerinde, belirtilen indisteki elemana veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiADXWilder için IND_ADXW).

CiBands

CiBands sınıfı Bollinger Bands® teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiBands sınıfı, Bollinger Bands® göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiBands: public CIndicator
```

Başlık

```
#include <Indicators\Trend.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiBands

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
MaShift	Yatay kaydırma değerine dönüş yapar
Deviation	Sapma değerine dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Base	Ana çizgi tamponunun elemanına dönüş yapar
Upper	Üst çizgi tamponunun elemanına dönüş yapar
Lower	Alt çizgi tamponunun elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

MaShift

Göstergenin yatay kaydırma değerine dönüş yapar.

```
int  MaShift() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş "yatay kaydırma" değerine dönüş yapar.

Deviation

Sapma değerine dönüş yapar.

```
double  Deviation() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan sapma değerine dönüş yapar.

Applied

Uygulanan fiyat türüne veya tanıtıcıya dönüş yapar

```
int  Applied() const
```

Dönüş değeri

Gösterge oluşturma sırasında tanımlanan fiyat türü veya tanıtıcı.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string            symbol,         // sembol
   ENUM_TIMEFRAMES   period,         // periyot
   int               ma_period,      // ortalama periyodu
   int               ma_shift,       // kaydırma değeri
   double            deviation,      // sapma
   int               applied         // fiyat tipi veya tanıtıcı değer
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

ma_shift

[in] Göstergenin yatay kaydırma değeri.

deviation

[in] Sapma değeri.

applied

[in] Uygulanacak hacim tipi.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Base

Ana çizginin tamponu üzerinde, belirtilen indisteki elemana dönüş yapar.

```
double  Base(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Ana çizginin tamponu üzerinde belirtilen indisteki elemana veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

Upper

Üst çizgi tamponunun belirtilen indisli elemanına dönüş yapar.

```
double  Upper(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Üst çizgi tamponunun belirtilen indisli elemanına veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

Lower

Alt çizgi tamponunun belirtilen indisli elemanına dönüş yapar.

```
double  Lower(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Ana çizginin tamponu üzerinde belirtilen indisteki elemana veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiBands için IND_BANDS).

CiEnvelopes

CiEnvelopes sınıfı, Envelopes teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiEnvelopes sınıfı, Envelopes göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiEnvelopes: public CIndicator
```

Başlık

```
#include <Indicators\Trend.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiEnvelopes

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
MaShift	Yatay kaydırma değerine dönüş yapar
MaMethod	Ortalama yöntemine dönüş yapar
Deviation	Sapma değerine dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Upper	Üst çizgi tamponunun elemanına dönüş yapar
Lower	Alt çizgi tamponunun elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

MaShift

Göstergenin yatay kaydırma değerine dönüş yapar.

```
int  MaShift() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş "yatay kaydırma" değerine dönüş yapar.

MaMethod

Ortalama yöntemine dönüş yapar.

```
ENUM_MA_METHOD  MaMethod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama türüne dönüş yapar.

Deviation

Sapma değerine dönüş yapar.

```
double  Deviation() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş "sapma" değerine dönüş yapar.

Applied

Uygulanan fiyat türüne veya tanıtıcıya dönüş yapar

```
int  Applied() const
```

Dönüş değeri

Gösterge oluşturma sırasında tanımlanan fiyat türü veya tanıtıcı.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,        // sembol
   ENUM_TIMEFRAMES  period,        // periyot
   int              ma_period,     // ortalama periyodu
   int              ma_shift,      // yatay kaydırma değeri
   ENUM_MA_METHOD   ma_method,     // ortalama yöntemi
   int              applied,       // uygulanan fiyat tipi veya tanıtıcı değer
   double           deviation      // sapma
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

ma_shift

[in] Yatay kaydırma değeri.

ma_method

[in] Ortalama periyodu (ENUM_MA_METHOD sayımının değerlerinden biri).

applied

[in] Uygulanan fiyat tipi veya tanıtıcı değer.

deviation

[in] Sapma değeri.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Upper

Üst çizgi tamponunun belirtilen indisli elemanına dönüş yapar.

```
double  Upper(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Üst çizgi tamponunun belirtilen indisli elemanına veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

Lower

Alt çizgi tamponunun belirtilen indisli elemanına dönüş yapar.

```
double  Lower(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Ana çizginin tamponu üzerinde belirtilen indisteki elemana veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiEnvelopes için IND_ENVELOPES).

CiIchimoku

CiIchimoku sınıfı, Ichimoku Kinko Hyo teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiIchimoku sınıfı, Ichimoku Kinko Hyo göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiIchimoku: public CIndicator
```

Başlık

```
#include <Indicators\Trend.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiIchimoku

Sınıf Yöntemleri

Özellikler	
TenkanSenPeriod	TenkanSen periyoduna dönüş yapar
KijunSenPeriod	KijunSen periyoduna dönüş yapar
SenkouSpanBPeriod	SenkouSpanB periyoduna dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
TenkanSen	TenkanSen çizgisi tamponunun elemanına dönüş yapar
KijunSen	KijunSen çizgisi tamponunun elemanına dönüş yapar
SenkouSpanA	SenkouSpanA çizgisi tamponunun elemanına dönüş yapar
SenkouSpanB	SenkouSpanB çizgisi tamponunun elemanına dönüş yapar

Özellikler	
ChikouSpan	ChikouSpan çizgisi tamponunun elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

TenkanSenPeriod

TenkanSen periyoduna dönüş yapar.

```
int  TenkanSenPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan TenkanSen periyoduna dönüş yapar.

KijunSenPeriod

KijunSen periyoduna dönüş yapar.

```
int  KijunSenPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan KijunSen periyodu.

SenkouSpanBPeriod

SenkouSpanB periyoduna dönüş yapar.

```
int  SenkouSpanBPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan SenkouSpanB periyodu.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,           // sembol
   ENUM_TIMEFRAMES  period,           // periyot
   int              tenkan_sen,       // TenkanSen periyodu
   int              kijun_sen,        // KijunSen periyodu
   int              senkou_span_b     // SenkouSpanB periyodu
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

tenkan_sen

[in] TenkanSen periyodu.

kijun_sen

[in] KijunSen periyodu.

senkou_span_b

[in] SenkouSpanB periyodu.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

TenkanSen

TenkanSen çizgisinin tamponu üzerinde, belirtilen indisteki elemana dönüş yapar.

```
double  TenkanSen(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

TenkanSen çizgisinin tamponu üzerinde belirtilen indisteki elemana veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

KijunSen

KijunSen çizgisinin tamponu üzerinde, belirtilen indisteki elemana dönüş yapar.

```
double  KijunSen(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

KijunSen çizgisinin tamponu üzerinde, belirtilen indisteki elemana veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

SenkouSpanA

SenkouSpanA çizgisinin tamponu üzerinde, belirtilen indisteki elemana dönüş yapar.

```
double   SenkouSpanA(
   int   index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

SenkouSpanA çizgisinin tamponu üzerinde, belirtilen indisteki elemana veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

SenkouSpanB

SenkouSpanB çizgisinin tamponu üzerinde, belirtilen indisteki elemana dönüş yapar.

```
double   SenkouSpanB(
   int   index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

SenkouSpanB çizgisinin tamponu üzerinde, belirtilen indisteki elemana veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

ChinkouSpan

ChinkouSpan çizgisinin tamponu üzerinde, belirtilen indisteki elemana dönüş yapar.

```
double  ChinkouSpan(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

ChinkouSpan çizgisinin tamponu üzerinde belirtilen indisteki elemana veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (Cilchimoku için IND_ICHIMOKU).

CiMA

CiMA sınıfı, Moving Average teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiMA sınıfı, Moving Average göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiMA: public CIndicator
```

Başlık

```
#include <Indicators\Trend.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiMA

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
MaShift	Yatay kaydırma değerine dönüş yapar
MaMethod	Ortalama yöntemine dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

MaShift

Göstergenin yatay kaydırma değerine dönüş yapar.

```
int  MaShift() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş "yatay kaydırma" değerine dönüş yapar.

MaMethod

Ortalama yöntemine dönüş yapar.

```
ENUM_MA_METHOD  MaMethod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama yöntemine (ENUM_MA_METHOD sayımının değerlerinden biri) dönüş yapar.

Applied

Uygulanan fiyat türüne veya tanıtıcıya dönüş yapar

```
int  Applied() const
```

Dönüş değeri

Gösterge oluşturma sırasında tanımlanan fiyat türü veya tanıtıcı.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string            string,        // sembol
   ENUM_TIMEFRAMES   period,        // periyot
   int               ma_period,     // ortalama periyodu
   int               ma_shift,      // yatay kaydırma değeri
   ENUM_MA_METHOD    ma_method,     // ortalama yöntemi
   int               applied        // uygulanan fiyat tipi veya tanıtıcı değer
   )
```

Parametreler

string

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

ma_shift

[in] Yatay kaydırma değeri.

ma_method

[in] Ortalama periyodu (ENUM_MA_METHOD sayımının değerlerinden biri).

applied

[in] Uygulanan fiyat tipi veya tanıtıcı değer.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiWPR için IND_MA).

CiSAR

CiSAR sınıfı, Parabolic Stop And Reverse System teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiSAR sınıfı, Parabolic Stop And Reverse System göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiSAR: public CIndicator
```

Başlık

```
#include <Indicators\Trend.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiSAR

Sınıf Yöntemleri

Özellikler	
SarStep	Hız artırım adımına dönüş yapar
Maximum	Fiyat takibi kat sayısına dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

SarStep

Hız artırım adımına dönüş yapar.

```
double  SarStep() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan hız artırım adımı değeri.

Maximum

Fiyat takibi kat sayısına dönüş yapar.

```
double  Maximum() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan fiyat takibi kat-sayısının değeri.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string            symbol,       // sembol
   ENUM_TIMEFRAMES   period,       // periyot
   double            step,         // adım
   double            maximum       // katsayı
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

step

[in] Hız artırımı için adım değeri.

maximum

[in] Fiyat takibi katsayısı.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiSAR için IND_SAR).

CiStdDev

CiStdDev sınıfı Standard Deviation teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiStdDev sınıfı Standard Deviation teknik göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiStdDev: public CIndicator
```

Başlık

```
#include <Indicators\Trend.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiStdDev

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
MaShift	Yatay kaydırma değerine dönüş yapar
MaMethod	Ortalama yöntemine dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

MaShift

Göstergenin yatay kaydırma değerine dönüş yapar.

```
int  MaShift() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş "yatay kaydırma" değerine dönüş yapar.

MaMethod

Ortalama yöntemine dönüş yapar.

```
ENUM_MA_METHOD  MaMethod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama yöntemine (ENUM_MA_METHOD sayımının değerlerinden biri) dönüş yapar.

Applied

Uygulanan fiyat türüne veya tanıtıcıya dönüş yapar

```
int  Applied() const
```

Dönüş değeri

Gösterge oluşturma sırasında tanımlanan fiyat türü veya tanıtıcı.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string            symbol,         // sembol
   ENUM_TIMEFRAMES   period,         // periyot
   int               ma_period,      // ortalama periyodu
   int               ma_shift,       // yatay kaydırma değeri
   ENUM_MA_METHOD    ma_method,      // ortalama yöntemi
   int               applied         // uygulanan fiyat tipi veya tanıtıcı değer
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

ma_shift

[in] Yatay kaydırma değeri.

ma_method

[in] Ortalama periyodu (ENUM_MA_METHOD sayımının değerlerinden biri).

applied

[in] Uygulanan fiyat tipi veya tanıtıcı değer.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiStdDev için IND_STDDEV).

CiDEMA

CiDEMA sınıfı, Double Exponential Moving Average teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiDEMA sınıfı, Double Exponential Moving Average göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiDEMA: public CIndicator
```

Başlık

```
#include <Indicators\Trend.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiDEMA

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
IndShift	Yatay kaydırma değerine dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

IndShift

Göstergenin yatay kaydırma değerine dönüş yapar.

```
int  IndShift() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş "yatay kaydırma" değerine dönüş yapar.

Applied

Uygulanan fiyat türüne veya tanıtıcıya dönüş yapar

```
int  Applied() const
```

Dönüş değeri

Gösterge oluşturma sırasında tanımlanan fiyat türü veya tanıtıcı.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           string,        // sembol
   ENUM_TIMEFRAMES  period,        // periyot
   int              ma_period,     // ortalama periyodu
   int              ind_shift,     // kaydırma değeri
   int              applied        // uygulanan fiyat tipi veya tanıtıcı değer
   )
```

Parametreler

string

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

ind_shift

[in] Yatay kaydırma değeri.

applied

[in] Uygulanan fiyat tipi veya tanıtıcı değer.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiDEMA için IND_DEMA).

CiTEMA

CiTEMA sınıfı, Triple Exponential Moving Average teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiTEMA sınıfı, Triple Exponential Moving Average göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiTEMA: public CIndicator
```

Başlık

```
#include <Indicators\Trend.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiTEMA

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
IndShift	Yatay kaydırma değerine dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

IndShift

Göstergenin yatay kaydırma değerine dönüş yapar.

```
int  IndShift() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş "yatay kaydırma" değerine dönüş yapar.

Applied

Uygulanan fiyat türüne veya tanıtıcıya dönüş yapar

```
int  Applied() const
```

Dönüş değeri

Gösterge oluşturma sırasında tanımlanan fiyat türü veya tanıtıcı.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,         // sembol
   ENUM_TIMEFRAMES  period,         // periyot
   int              ma_period,      // ortalama periyodu
   int              ma_shift,       // girinti
   int              applied         // uygulanan fiyat tipi veya tanıtıcı değer
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

ma_shift

[in] Yatay kaydırma değeri.

applied

[in] Uygulanan fiyat tipi veya tanıtıcı değer.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiTEMA için IND_TEMA).

CiFrAMA

CiFrAMA sınıfı, Fractal Adaptive Moving Average teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiFrAMA sınıfı, Fractal Adaptive Moving Average göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiFrAMA: public CIndicator
```

Başlık

```
#include <Indicators\Trend.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiFrAMA

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
IndShift	Yatay kaydırma değerine dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

IndShift

Göstergenin yatay kaydırma değerine dönüş yapar.

```
int  IndShift() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş "yatay kaydırma" değerine dönüş yapar.

Applied

Uygulanan fiyat türüne veya tanıtıcıya dönüş yapar

```
int  Applied() const
```

Dönüş değeri

Gösterge oluşturma sırasında tanımlanan fiyat türü veya tanıtıcı.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,        // sembol
   ENUM_TIMEFRAMES  period,        // periyot
   int              ma_period,     // ortalama periyodu
   int              ma_shift,      // girinti
   int              applied        // uygulanan fiyat tipi veya tanıtıcı değer
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

ma_shift

[in] Yatay kaydırma değeri.

applied

[in] Uygulanan fiyat tipi veya tanıtıcı değer.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiFrAMA için IND_FRAMA).

CiAMA

CiAMA sınıfı, Adaptive Moving Average teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiAMA sınıfı, Adaptive Moving Average göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiAMA: public CIndicator
```

Başlık

```
#include <Indicators\Trend.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAMA

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
FastEmaPeriod	Hızlı EMA için ortalama periyoduna dönüş yapar
SlowEmaPeriod	Yavaş EMA için ortalama periyoduna dönüş yapar
IndShift	Yatay kaydırma değerine dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

FastEmaPeriod

Hızlı EMA için ortalama periyoduna dönüş yapar.

```
int  FastEmaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında tanımlanan hızlı EMA periyoduna dönüş yapar.

SlowEmaPeriod

Yavaş EMA için ortalama periyoduna dönüş yapar.

```
int  SlowEmaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında tanımlanan yavaş EMA periyoduna dönüş yapar.

IndShift

Göstergenin yatay kaydırma değerine dönüş yapar.

```
int  IndShift() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş "yatay kaydırma" değerine dönüş yapar.

Applied

Uygulanan fiyat türüne veya tanıtıcıya dönüş yapar

```
int  Applied() const
```

Dönüş değeri

Gösterge oluşturma sırasında tanımlanan fiyat türü veya tanıtıcı.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           string,              // sembol
   ENUM_TIMEFRAMES  period,              // periyot
   int              ma_period,           // ortalama periyodu
   int              fast_ema_period,     // hızlı EMA Periyodu
   int              slow_ema_period,     // yavaş EMA periyodu
   int              ind_shift,           // kaydırma değeri
   int              applied              // uygulanan fiyat tipi veya tanıtıcı değer
   )
```

Parametreler

string

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

fast_ema_period

[in] Hızlı EMA periyodu.

slow_ema_period

[in] Yavaş EMA periyodu.

ind_shift

[in] Yatay kaydırma değeri.

applied

[in] Uygulanan fiyat tipi veya tanıtıcı değer.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiAMA için IND_AMA).

CiVIDyA

CiVIDyA sınıfı, Variable Index Dynamic Average teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiVIDyA sınıfı, Variable Index Dynamic Average göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiVIDyA: public CIndicator
```

Başlık

```
#include <Indicators\Trend.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiVIDyA

Sınıf Yöntemleri

Özellikler	
CmoPeriod	Momentum periyoduna dönüş yapar.
EmaPeriod	EMA için ortalama periyoduna dönüş yapar
IndShift	Yatay kaydırma değerine dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

CmoPeriod

Momentum periyoduna dönüş yapar.

```
int  CmoPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş Momentum periyoduna dönüş yapar.

EmaPeriod

EMA için ortalama periyoduna dönüş yapar.

```
int  EmaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında tanımlanan EMA periyoduna dönüş yapar.

IndShift

Göstergenin yatay kaydırma değerine dönüş yapar.

```
int  IndShift() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş "yatay kaydırma" değerine dönüş yapar.

Applied

Uygulanan fiyat türüne veya tanıtıcıya dönüş yapar

```
int  Applied() const
```

Dönüş değeri

Gösterge oluşturma sırasında tanımlanan fiyat türü veya tanıtıcı.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,          // sembol
   ENUM_TIMEFRAMES  period,          // periyot
   int              cmo_period,      // Momentum periyodu
   int              ema_period,      // ortalama periyodu
   int              ind_shift,       // kaydırma değeri
   int              applied          // uygulanan fiyat tipi veya tanıtıcı değer
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

cmo_period

[in] Momentum periyodu.

ema_period

[in] Ortalama periyodu.

ind_shift

[in] Yatay kaydırma değeri.

applied

[in] Uygulanan fiyat tipi veya tanıtıcı değer.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiVIDyA için IND_VIDYA).

Osilatörler

Bu bölüm, Osilatör sınıflarıyla çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için açıklamalar içermektedir.

Sınıf/grup	Açıklama
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 sınıfı, Average True Range teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiATR sınıfı, Average True Range göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiATR: public CIndicator
```

Başlık

```
#include <Indicators\Oscilators.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiATR

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,        // sembol
   ENUM_TIMEFRAMES  period,        // periyot
   int              ma_period      // ortalama periyodu
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nene tipi tanımlayıcısı (CiATR için IND_ATR).

CiBearsPower

CiBearsPower sınıfı, Bears Power teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiBearsPower, Bears Power göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiBearsPower: public CIndicator
```

Başlık

```
#include <Indicators\Oscilators.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiBearsPower

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CArrayObj

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string            symbol,         // sembol
   ENUM_TIMEFRAMES   period,         // periyot
   int               ma_period       // ortalama periyodu
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiBearsPower için IND_BEARS).

CiBullsPower

CiBullsPower, Bulls Power teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiBullsPower, Bulls Power göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiBullsPower: public CIndicator
```

Başlık

```
#include <Indicators\Oscilators.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiBullsPower

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CArrayObj

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string            symbol,         // sembol
   ENUM_TIMEFRAMES   period,         // periyot
   int               ma_period       // ortalama periyodu
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CiBullsPower için IND_BULLS).

CiCCI

CiCCI sınıfı, Commodity Channel Index teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiCCI sınıfı, Commodity Channel Index göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiCCI: public CIndicator
```

Başlık

```
#include <Indicators\Oscilators.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiCCI

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

Applied

Uygulanan fiyat türüne veya tanıtıcıya dönüş yapar

```
int  Applied() const
```

Dönüş değeri

Gösterge oluşturma sırasında tanımlanan fiyat türü veya tanıtıcı.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string            symbol,         // sembol
   ENUM_TIMEFRAMES   period,         // periyot
   int               ma_period,      // ortalama periyodu
   int               applied         // uygulanan fiyat tipi veya tanıtıcı değer
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

applied

[in] Uygulanan fiyat tipi veya tanıtıcı değer.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CiCCI için IND_CCI).

CiChaikin

CiChaikin sınıfı, Chaikin Oscillator teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiChaikin sınıfı, Chaikin Oscillator teknik göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiChaikin: public CIndicator
```

Başlık

```
#include <Indicators\Oscilators.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiChaikin

Sınıf Yöntemleri

Özellikler	
FastMaPeriod	Hızlı MA (hareketli ortalama) için ortalama periyoduna dönüş yapar
SlowMaPeriod	Yavaş MA için ortalama periyoduna dönüş yapar
MaMethod	Ortalama yöntemine dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

FastMaPeriod

Hızlı EMA için ortalama periyoduna dönüş yapar.

```
int  FastMaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında tanımlanan hızlı EMA periyoduna dönüş yapar.

SlowMaPeriod

Yavaş EMA için ortalama periyoduna dönüş yapar.

```
int  SlowMaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında tanımlanan yavaş EMA periyoduna dönüş yapar.

MaMethod

Ortalama yöntemine dönüş yapar.

```
ENUM_MA_METHOD  MaMethod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama türüne dönüş yapar.

Applied

Uygulanacak hacim tipine dönüş yapar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Dönüş değeri

Oluşturma sırasında tanımlanan - uygulanacak - hacim tipi.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string               symbol,              // sembol
   ENUM_TIMEFRAMES      period,              // periyot
   int                  fast_ma_period,      // hızlı EMA periyodu
   int                  slow_ma_period,      // yavaş EMA periyodu
   ENUM_MA_METHOD       ma_method,           // ortalama yöntemi
   ENUM_APPLIED_VOLUME  applied              // uygulanacak hacim tipi
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

fast_ma_period

[in] Hızlı EMA periyodu.

slow_ma_period

[in] Yavaş EMA periyodu.

ma_method

[in] Ortalama periyodu (ENUM_MA_METHOD sayımının değerlerinden biri).

applied

[in] Uygulanacak hacim (ENUM_APPLIED_VOLUME sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CiChaikiniçin IND_CHAIKIN).

CiDeMarker

CiDeMarker sınıfı, DeMarker teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiDeMarker sınıfı, DeMarker göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiDeMarker: public CIndicator
```

Başlık

```
#include <Indicators\Oscilators.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiDeMarker

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CArrayObj

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,        // sembol
   ENUM_TIMEFRAMES  period,        // periyot
   int              ma_period      // ortalama periyodu
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiDeMarker için IND_DEMARKER).

CiForce

CiForce sınıfı, Force Index teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiForce sınıfı, Force Index göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiForce: public CIndicator
```

Başlık

```
#include <Indicators\Oscilators.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiForce

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
MaMethod	Ortalama yöntemine dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

MaMethod

Ortalama yöntemine dönüş yapar.

```
ENUM_MA_METHOD  MaMethod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama türüne dönüş yapar.

Applied

Uygulanacak hacim tipine dönüş yapar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Dönüş değeri

Oluşturma sırasında tanımlanan - uygulanacak - hacim tipi.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string               symbol,        // sembol
   ENUM_TIMEFRAMES      period,        // periyot
   int                  ma_period,     // ortalama periyodu
   ENUM_MA_METHOD       ma_method,     // ortalama yöntemi
   ENUM_APPLIED_VOLUME  applied        // uygulanan hacim tipi
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

ma_method

[in] Ortalama periyodu (ENUM_MA_METHOD sayımının değerlerinden biri).

applied

[in] Uygulanacak hacim (ENUM_APPLIED_VOLUME sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CiForce için IND_FORCE).

CiMACD

CiMACD sınıfı, Moving Averages Convergence-Divergence teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiMACD sınıfı, Moving Averages Convergence-Divergence göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiMACD: public CIndicator
```

Başlık

```
#include <Indicators\Oscilators.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiMACD

Sınıf Yöntemleri

Özellikler	
FastEmaPeriod	Hızlı EMA için ortalama periyoduna dönüş yapar
SlowEmaPeriod	Yavaş EMA için ortalama periyoduna dönüş yapar
SignalPeriod	Sinyal çizgisinin ortalama periyoduna dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Ana çizgi tamponunun elemanına dönüş yapar
Signal	Sinyal çizgisi tamponunun elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

FastEmaPeriod

Hızlı EMA için ortalama periyoduna dönüş yapar.

```
int  FastEmaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında tanımlanan hızlı EMA periyoduna dönüş yapar.

SlowEmaPeriod

Yavaş EMA için ortalama periyoduna dönüş yapar.

```
int  SlowEmaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında tanımlanan yavaş EMA periyoduna dönüş yapar.

SignalPeriod

Sinyal çizgisinin ortalama periyoduna dönüş yapar.

```
int  SignalPeriod() const
```

Dönüş değeri

Sinyal çizgisinin – göstergenin oluşturulması sırasında belirtilmiş olan – ortalama periyoduna dönüş yapar.

Applied

Uygulanan fiyat türüne veya tanıtıcıya dönüş yapar

```
int  Applied() const
```

Dönüş değeri

Gösterge oluşturma sırasında tanımlanan fiyat türü veya tanıtıcı.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,              // sembol
   ENUM_TIMEFRAMES  period,              // periyot
   int              fast_ema_period,     // hızlı EMA Periyodu
   int              slow_ema_period,     // yavaş EMA periyodu
   int              signal_period,       // sinyal periyodu
   int              applied              // uygulanan fiyat tipi veya tanıtıcı değer
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

fast_ema_period

[in] Hızlı EMA periyodu.

slow_ema_period

[in] Yavaş EMA periyodu.

signal_period

[in] Sinyal çizgisi periyodu.

applied

[in] Uygulanan fiyat tipi veya tanıtıcı değer.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Signal

Sinyal çizgisinin tamponu üzerinde, belirtilen indisteki elemana dönüş yapar.

```
double  Signal(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Sinyal çizgisinin tamponu üzerinde belirtilen indisteki elemana veya geçerli veri yoksaEMPTY_VALUE değerine dönüş yapar.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CiMACD için IND_MACD).

CiMomentum

CiMomentum sınıfı, Momentum teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiMomentum sınıfı, Momentum göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiMomentum: public CIndicator
```

Başlık

```
#include <Indicators\Oscilators.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiMomentum

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

Applied

Uygulanan fiyat türüne veya tanıtıcıya dönüş yapar

```
int  Applied() const
```

Dönüş değeri

Gösterge oluşturma sırasında tanımlanan fiyat türü veya tanıtıcı.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,         // sembol
   ENUM_TIMEFRAMES  period,         // periyot
   int              ma_period,      // ortalama periyodu
   int              applied         // uygulanan fiyat tipi veya tanıtıcı değer
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

applied

[in] Uygulanan fiyat tipi veya tanıtıcı değer.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CiMomentum için IND_MOMENTUM).

CiOsMA

CiOsMA sınıfı, Moving Average of Oscillator (MACD histogram) teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiOsMA class provides the creation, setup and access to the data of the Moving Average of Oscillator (MACD histogram) indicator.

Bildirim

```
class CiOsMA: public CIndicator
```

Başlık

```
#include <Indicators\Oscilators.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiOsMA

Sınıf Yöntemleri

Özellikler	
FastEmaPeriod	Hızlı EMA için ortalama periyoduna dönüş yapar
SlowEmaPeriod	Yavaş EMA için ortalama periyoduna dönüş yapar
SignalPeriod	Sinyal çizgisinin ortalama periyoduna dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

FastEmaPeriod

Hızlı EMA için ortalama periyoduna dönüş yapar.

```
int  FastEmaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında tanımlanan hızlı EMA periyoduna dönüş yapar.

SlowEmaPeriod

Yavaş EMA için ortalama periyoduna dönüş yapar.

```
int  SlowEmaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında tanımlanan yavaş EMA periyoduna dönüş yapar.

SignalPeriod

Sinyal çizgisinin ortalama periyoduna dönüş yapar.

```
int  SignalPeriod() const
```

Dönüş değeri

Sinyal çizgisinin – göstergenin oluşturulması sırasında belirtilmiş olan – ortalama periyoduna dönüş yapar.

Applied

Uygulanan fiyat türüne veya tanıtıcıya dönüş yapar

```
int  Applied() const
```

Dönüş değeri

Gösterge oluşturma sırasında tanımlanan fiyat türü veya tanıtıcı.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,              // sembol
   ENUM_TIMEFRAMES  period,              // periyot
   int              fast_ema_period,     // hızlı EMA Periyodu
   int              slow_ema_period,     // yavaş EMA periyodu
   int              signal_period,       // sinyal çizgisi periyodu
   int              applied              // uygulanan fiyat tipi veya tanıtıcı değer
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

fast_ema_period

[in] Hızlı EMA periyodu.

slow_ema_period

[in] Yavaş EMA periyodu.

signal_period

[in] Sinyal çizgisi periyodu.

applied

[in] Uygulanan fiyat tipi veya tanıtıcı değer.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CiOsMA için IND_OSMA).

CiRSI

CiRSI sınıfı, Relative Strength Index teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiRSI sınıfı, Relative Strength Index göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiRSI: public CIndicator
```

Başlık

```
#include <Indicators\Oscilators.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiRSI

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

Applied

Uygulanan fiyat türüne veya tanıtıcıya dönüş yapar

```
int  Applied() const
```

Dönüş değeri

Gösterge oluşturma sırasında tanımlanan fiyat türü veya tanıtıcı.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,         // sembol
   ENUM_TIMEFRAMES  period,         // periyot
   int              ma_period,      // ortalama periyodu
   int              applied         // uygulanan fiyat tipi veya tanıtıcı değer
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

applied

[in] Uygulanan fiyat tipi veya tanıtıcı değer.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CiRSI için IND_RSI).

CiRVI

CiRVI sınıfı, Relative Vigor Index teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiRVI sınıfı, Relative Vigor Index göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiRVI: public CIndicator
```

Başlık

```
#include <Indicators\Oscilators.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiRVI

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Ana çizgi tamponunun elemanına dönüş yapar
Signal	Sinyal çizgisi tamponunun elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string            symbol,         // sembol
   ENUM_TIMEFRAMES   period,         // periyot
   int               ma_period       // ortalama periyodu
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Signal

Sinyal çizgisinin tamponu üzerinde, belirtilen indisteki elemana dönüş yapar.

```
double  Signal(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Sinyal çizgisinin tamponu üzerinde belirtilen indisteki elemana veya geçerli veri yoksaEMPTY_VALUE değerine dönüş yapar.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CiRVI için IND_RVI).

CiStochastic

CiStochastic sınıfı Stochastic Oscillator teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiStochastic sınıfı Stochastic Oscillator teknik göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiStochastic: public CIndicator
```

Başlık

```
#include <Indicators\Oscilators.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiStochastic

Sınıf Yöntemleri

Özellikler	
Kperiod	%K çizgisinin ortalama periyoduna dönüş yapar
Dperiod	%D çizgisinin ortalama periyoduna dönüş yapar
Slowing	Yavaşlama periyoduna dönüş yapar
MaMethod	Ortalama yöntemine dönüş yapar
PriceField	Uygulanacak (Low/High veya Close/Close) fiyat tipi
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Ana çizgi tamponunun elemanına dönüş yapar
Signal	Sinyal çizgisi tamponunun elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

Kperiod

%K çizgisinin ortalama periyoduna dönüş yapar.

```
int  Kperiod() const
```

Dönüş değeri

Oluşturma sırasında tanımlanan %K çizgisinin ortalama periyoduna dönüş yapar.

Dperiod

%D çizgisinin ortalama periyoduna dönüş yapar.

```
int  Dperiod() const
```

Dönüş değeri

Oluşturma sırasında tanımlanan %D çizgisinin ortalama periyoduna dönüş yapar.

Slowing

Yavaşlama periyoduna dönüş yapar.

```
int  Slowing() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş "yavaşlama periyodu" değerine dönüş yapar.

MaMethod

Ortalama yöntemine dönüş yapar.

```
ENUM_MA_METHOD  MaMethod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama türüne dönüş yapar.

PriceField

Uygulanacak (Low/High veya Close/Close) fiyat tipi.

```
ENUM_STO_PRICE  PriceField() const
```

Dönüş değeri

Oluşturma sırasında tanımlanan - uygulanacak - fiyat tipi.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,          // sembol
   ENUM_TIMEFRAMES  period,          // periyot
   int              Kperiod,         // %K ortalama periyodu
   int              Dperiod,         // %D ortalama periyodu
   int              slowing,         // Yavaşlama periyodu
   ENUM_MA_METHOD   ma_method,       // ortalama yöntemi
   ENUM_STO_PRICE   price_field      // Uygulanacak fiyat tpi
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

Kperiod

[in] %K çizgisinin ortalama periyodu.

Dperiod

[in] %D çizgisinin ortalama periyodu.

slowing

[in] Yavaşlama periyodu.

ma_method

[in] Ortalama periyodu (ENUM_MA_METHOD sayımının değerlerinden biri).

price_field

[in] Uygulanacak fiyat tipi (Low/High veya Close/Close) (ENUM_STO_PRICE sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Signal

Sinyal çizgisinin tamponu üzerinde, belirtilen indisteki elemana dönüş yapar.

```
double  Signal(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Signal

Sinyal çizgisinin tamponu üzerinde, belirtilen indisteki elemana dönüş yapar.

```
double  Signal(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Sinyal çizgisinin tamponu üzerinde belirtilen indisteki elemana veya geçerli veri yoksaEMPTY_VALUE değerine dönüş yapar.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CiStochastic için IND_STOCHASTIC).

CiTriX

CiTriX sınıfı, Triple Exponential Moving Averages Oscillator teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiTriX sınıfı, Triple Exponential Moving Averages Oscillator göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiTriX: public CIndicator
```

Başlık

```
#include <Indicators\Oscilators.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiTriX

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

Applied

Uygulanan fiyat türüne veya tanıtıcıya dönüş yapar

```
int  Applied() const
```

Dönüş değeri

Gösterge oluşturma sırasında tanımlanan fiyat türü veya tanıtıcı.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,         // sembol
   ENUM_TIMEFRAMES  period,         // periyot
   int              ma_period,      // ortalama periyodu
r   int              applied         // fiyat tipi veya tanıtıcı değer
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

applied

[in] Uygulanan fiyat tipi veya tanıtıcı değer.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Object type identifier (IND_TRIX for CiTriX).

CiWPR

CiWPR sınıfı, Williams' Percent Range teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiWPR sınıfı, Williams' Percent Range göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiWPR: public CIndicator
```

Başlık

```
#include <Indicators\Oscilators.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiWPR

Sınıf Yöntemleri

Özellikler	
CalcPeriod	Hesaplama periyoduna dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

CalcPeriod

Hesaplama periyoduna dönüş yapar.

```
int  CalcPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş "hesaplama periyodu" değerine dönüş yapar.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,          // sembol
   ENUM_TIMEFRAMES  period,          // periyot
   int              calc_period      // hesaplama periyodu
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

calc_period

[in] Hesaplama periyodu.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiWPR için IND_WPR).

Hacim göstergeleri

Bu bölüm, Hacim göstergelerinin sınıflarıyla çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için açıklamalar içermektedir.

Sınıf/grup	Açıklama
CiAD	Accumulation/Distribution
CiMFI	Money Flow Index
CiOBV	On Balance Volume
CiVolumes	Volumes

CiAD

CiAD sınıfı Accumulation/Distribution teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiAD sınıfı, Accumulation/Distribution göstergesi için oluşturma, başlatma ve veri erişimi gibi yöntemler sağlar.

Bildirim

```
class CiAD: public CIndicator
```

Başlık

```
#include <Indicators\Volumes.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAD

Sınıf Yöntemleri

Özellikler	
Applied	Uygulanacak hacim tipine dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

Applied

Uygulanacak hacim tipine dönüş yapar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Dönüş değeri

Oluşturma sırasında tanımlanan - uygulanacak - hacim tipi.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string               symbol,      // sembol
   ENUM_TIMEFRAMES      period,      // periyot
   ENUM_APPLIED_VOLUME  applied      // uygulanacak hacim
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

applied

[in] Uygulanacak hacim tipi (ENUM_APPLIED_VOLUME sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiAD için IND_AD).

CiMFI

CiMFI sınıfı, Money Flow Index teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiMFI sınıfı, Money Flow Index göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiMFI: public CIndicator
```

Başlık

```
#include <Indicators\Volumes.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiMFI

Sınıf Yöntemleri

Özellikler	
MaPeriod	Ortalama periyoduna dönüş yapar
Applied	Uygulanacak hacim tipine dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

MaPeriod

Ortalama periyoduna dönüş yapar.

```
int  MaPeriod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama periyoduna dönüş yapar.

Applied

Uygulanacak hacim tipine dönüş yapar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Dönüş değeri

Oluşturma sırasında tanımlanan - uygulanacak - hacim tipi.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string               symbol,         // sembol
   ENUM_TIMEFRAMES      period,         // periyot
   int                  ma_period,      // ortalama periyodu
   ENUM_APPLIED_VOLUME  applied         // uygulanan hacim tipi
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

ma_period

[in] Ortalama periyodu.

applied

[in] Uygulanacak hacim tipi (ENUM_APPLIED_VOLUME sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Object type identifier (IND_MFI for CiMFI).

CiOBV

CiOBV sınıfı On Balance Volume teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiOBV sınıfı On Balance Volume teknik göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiOBV: public CIndicator
```

Başlık

```
#include <Indicators\Volumes.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiOBV

Sınıf Yöntemleri

Özellikler	
Applied	Uygulanacak hacim tipine dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

Applied

Uygulanacak hacim tipine dönüş yapar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Dönüş değeri

Oluşturma sırasında tanımlanan - uygulanacak - hacim tipi.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string               symbol,      // sembol
   ENUM_TIMEFRAMES      period,      // periyot
   ENUM_APPLIED_VOLUME  applied      // uygulanacak hacim
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

applied

[in] Uygulanacak hacim tipi (ENUM_APPLIED_VOLUME sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiOBV için IND_OBV).

CiVolumes

CiVolumes sınıfı Volumes teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiVolumes sınıfı Volumes teknik göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiVolumes: public CIndicator
```

Başlık

```
#include <Indicators\Volumes.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiVolumes

Sınıf Yöntemleri

Özellikler	
Applied	Uygulanacak hacim tipine dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

Applied

Uygulanacak hacim tipine dönüş yapar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Dönüş değeri

Oluşturma sırasında tanımlanan - uygulanacak - hacim tipi.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string               symbol,      // sembol
   ENUM_TIMEFRAMES      period,      // periyot
   ENUM_APPLIED_VOLUME  applied      // uygulanacak hacim
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

applied

[in] Uygulanacak hacim tipi (ENUM_APPLIED_VOLUME sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcı (CiVolumes için IND_VOLUMES).

Bill Williams Göstergeleri

Bu bölüm, Bill Williams göstergelerinin sınıflarıyla çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için açıklamalar içermektedir.

Sınıf/grup	Açıklama
CiAC	"Accelerator Oscillator" göstergesi
CiAlligator	"Alligator" göstergesi
CiAO	"Awesome Oscillator" göstergesi
CiFractals	"Fractals" göstergesi
CiGator	"Gator Oscillator" göstergesi
CiBWMFI	"Market Facilitation Index" göstergesi

CiAC

CiAC sınıfı, Accelerator Oscillator göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiAC sınıfı, Accelerator Oscillator göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiAC: public CIndicator
```

Başlık

```
#include <Indicators\BillWilliams.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAC

Sınıf Yöntemleri

Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,      // Sembol
   ENUM_TIMEFRAMES  period       // Periyot
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CiAC için IND_AC).

CiAlligator

CiAlligator sınıfı, Alligator teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiAlligator sınıfı, Alligator göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiAlligator: public CIndicator
```

Başlık

```
#include <Indicators\BillWilliams.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAlligator

Sınıf Yöntemleri

Özellikler	
JawPeriod	Jaw (çene) çizgisinin ortalama periyoduna dönüş yapar
JawShift	Jaw çizgisinin yatay kaydırma değerine dönüş yapar
TeethPeriod	Teeth (diş) çizgisinin ortalama periyodunu alır
TeethShift	Teeths çizgisinin yatay kaydırma değerine dönüş yapar
LipsPeriod	Lips (dudak) çizgisinin ortalama periyodunu alır
LipsShift	Lips çizgisinin yatay kaydırma değerine dönüş yapar
MaMethod	Ortalama yöntemine dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Jaw	Jaws çizgisi tamponunun elemanına dönüş yapar

Özellikler	
Teeth	Teeths çizgisi tamponunun elemanına dönüş yapar
Lips	Lips çizgisi tamponunun elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

JawPeriod

Jaw (çene) çizgisinin ortalama periyoduna dönüş yapar.

```
int  JawPeriod() const
```

Dönüş değeri

Jaw çizgisinin – göstergenin oluşturulması sırasında belirtilmiş olan – ortalama periyoduna dönüş yapar.

JawShift

Jaws (çene) çizgisinin yatay kaydırma değerine dönüş yapar.

```
int  JawShift() const
```

Dönüş değeri

Jaws çizgisinin - göstergenin oluşturulması sırasında belirtilmiş olan - yatay kaydırma değerine dönüş yapar.

TeethPeriod

Teeth (diş) çizgisinin ortalama periyodunu alır.

```
int  TeethPeriod() const
```

Dönüş değeri

Teeth çizgisinin – göstergenin oluşturulması sırasında belirtilmiş olan – ortalama periyoduna dönüş yapar.

TeethShift

Teeths (diş) çizgisinin yatay kaydırma değerine dönüş yapar

```
int  TeethShift() const
```

Dönüş değeri

Teeths çizgisinin – göstergenin oluşturulması sırasında belirtilmiş olan – yatay kaydırma değerine dönüş yapar.

LipsPeriod

Lips (dudak) çizgisinin ortalama periyodunu alır.

```
int  LipsPeriod() const
```

Dönüş değeri

Lips çizgisinin – göstergenin oluşturulması sırasında belirtilmiş olan – ortalama periyoduna dönüş yapar.

LipsShift

Lips (dudak) çizgisinin yatay kaydırma değerine dönüş yapar.

```
int  LipsShift() const
```

Dönüş değeri

Lips çizgisinin - göstergenin oluşturulması sırasında belirtilmiş olan - yatay kaydırma değerine dönüş yapar.

MaMethod

Ortalama yöntemine dönüş yapar.

```
ENUM_MA_METHOD  MaMethod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama türüne dönüş yapar.

Applied

Uygulanan fiyat türüne veya tanıtıcıya dönüş yapar

```
int  Applied() const
```

Dönüş değeri

Gösterge oluşturma sırasında tanımlanan fiyat türü veya tanıtıcı.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,           // Sembol
   ENUM_TIMEFRAMES  period,           // Periyot
   int              jaw_period,       // Jaws (çene) periyodu
   int              jaw_shift,        // Jaws kaydırma değeri
   int              teeth_period,     // Teeths (diş) periyodu
   int              teeth_shift,      // Teeths kaydırma değeri
   int              lips_period,      // Lips (dudak) periyodu
   int              lips_shift,       // Lips kaydırma değeri
   ENUM_MA_METHOD   ma_method,        // Ortalama türü
   int              applied           // uygulanacak hacim türü
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

jaw_period

[in] Jaws (çene) periyodu.

jaw_shift

[in] Jaws kaydırma değeri.

teeth_period

[in] Teeths (diş) periyodu.

teeth_shift

[in] Teeths kaydırma değeri.

lips_period

[in] Lips (dudak) periyodu.

lips_shift

[in] Lips kaydırma değeri.

ma_method

[in] Hareketli ortalama türü (ENUM_MA_METHOD sayımının değerlerinden biri).

applied

[in] Uygulanacak hacim tipi.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Jaw

Jaws (çene) çizgisinin tamponu üzerinde, belirtilen indisteki elemana dönüş yapar.

```
double   Jaw(
   int   index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Jaws çizgisinin tamponu üzerinde belirtilen indisteki elemana veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

Teeth

Teeths (diş) çizgisinin tamponu üzerinde, belirtilen indisteki elemana dönüş yapar.

```
double  Teeth(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Teeths çizgisinin tamponu üzerinde belirtilen indisteki elemana veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

Lips

Lips (dudak) çizgisinin tamponu üzerinde, belirtilen indisteki elemana dönüş yapar.

```
double  Lips(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Lips çizgisinin tamponu üzerinde belirtilen indisteki elemana veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CiAlligator için IND_ALLIGATOR).

CiAO

CiAO sınıfı, Awesome Oscillator teknik göstergesini kullanmak için geliştirilmiştir.

Açıklama

CiAO sınıfı, Awesome Oscillator göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiAO: public CIndicator
```

Başlık

```
#include <Indicators\BillWilliams.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiAO

Sınıf Yöntemleri

Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CArrayObj

FreeMode, FreeMode, Save, Load, CreateElement, Reserve, Resize, Shutdown, Add, AddArray, Insert, InsertArray, AssignArray, At, Update, Shift, Detach, Delete, DeleteRange, Clear,

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

CompareArray, InsertSort, Search, SearchGreat, SearchLess, SearchGreatOrEqual, SearchLessOrEqual, SearchFirst, SearchLast

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,      // Sembol
   ENUM_TIMEFRAMES  period       // Periyot
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CiAO için IND_AO).

CiFractals

CiFractals sınıfı, Fractals teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiFractals sınıfı Fraktals göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiFractals: public CIndicator
```

Başlık

```
#include <Indicators\BillWilliams.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiFractals

Sınıf Yöntemleri

Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Upper	Üst tamponun elemanına dönüş yapar
Lower	Alt tamponun elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CArrayObj

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,      // Sembol
   ENUM_TIMEFRAMES  period       // Periyot
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Upper

Üst tamponun belirtilen indisli elemanına dönüş yapar.

```
double  Upper(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Üst tamponun belirtilen indisli elemanına veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

Lower

Alt tamponun belirtilen indisli elemanına dönüş yapar.

```
double  Lower(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Alt tamponun belirtilen indisli elemanına veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CiFractals için IND_FRACTALS).

CiGator

CiGator sınıfı Gator Oscillator teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiGator sınıfı Gator Oscillator teknik göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiGator: public CIndicator
```

Başlık

```
#include <Indicators\BillWilliams.mqh>
```

Kalıtım hiyerarşisi

CObject
  CArray
    CArrayObj
      CSeries
        CIndicator
          CiGator

Sınıf Yöntemleri

Özellikler	
JawPeriod	Jaw (çene) çizgisinin ortalama periyoduna dönüş yapar
JawShift	Jaw çizgisinin yatay kaydırma değerine dönüş yapar
TeethPeriod	Teeth (diş) çizgisinin ortalama periyodunu alır
TeethShift	Teeths çizgisinin yatay kaydırma değerine dönüş yapar
LipsPeriod	Lips (dudak) çizgisinin ortalama periyodunu alır
LipsShift	Lips çizgisinin yatay kaydırma değerine dönüş yapar
MaMethod	Ortalama yöntemine dönüş yapar
Applied	Uygulanan fiyat türüne dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Upper	Üst tamponun elemanına dönüş yapar

Özellikler	
Lower	Alt tamponun elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

JawPeriod

Jaw (çene) çizgisinin ortalama periyoduna dönüş yapar.

```
int  JawPeriod() const
```

Dönüş değeri

Jaw çizgisinin – göstergenin oluşturulması sırasında belirtilmiş olan – ortalama periyoduna dönüş yapar.

JawShift

Jaws (çene) çizgisinin yatay kaydırma değerine dönüş yapar.

```
int  JawShift() const
```

Dönüş değeri

Jaws çizgisinin - göstergenin oluşturulması sırasında belirtilmiş olan - yatay kaydırma değerine dönüş yapar.

TeethPeriod

Teeth (diş) çizgisinin ortalama periyodunu alır.

```
int  TeethPeriod() const
```

Dönüş değeri

Teeth çizgisinin – göstergenin oluşturulması sırasında belirtilmiş olan – ortalama periyoduna dönüş yapar.

TeethShift

Teeths (diş) çizgisinin yatay kaydırma değerine dönüş yapar

```
int  TeethShift() const
```

Dönüş değeri

Teeths çizgisinin - göstergenin oluşturulması sırasında belirtilmiş olan - yatay kaydırma değerine dönüş yapar.

LipsPeriod

Lips (dudak) çizgisinin ortalama periyodunu alır.

```
int  LipsPeriod() const
```

Dönüş değeri

Lips çizgisinin – göstergenin oluşturulması sırasında belirtilmiş olan – ortalama periyoduna dönüş yapar.

LipsShift

Lips (dudak) çizgisinin yatay kaydırma değerine dönüş yapar.

```
int  LipsShift() const
```

Dönüş değeri

Lips çizgisinin - göstergenin oluşturulması sırasında belirtilmiş olan - yatay kaydırma değerine dönüş yapar.

MaMethod

Ortalama yöntemine dönüş yapar.

```
ENUM_MA_METHOD  MaMethod() const
```

Dönüş değeri

Göstergenin oluşturulması sırasında belirtilmiş olan ortalama türüne dönüş yapar.

Applied

Uygulanan fiyat türüne veya tanıtıcıya dönüş yapar

```
int  Applied() const
```

Dönüş değeri

Gösterge oluşturma sırasında tanımlanan fiyat türü veya tanıtıcı.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string           symbol,           // Sembol
   ENUM_TIMEFRAMES  period,           // Periyot
   int              jaw_period,       // Jaws (çene) periyodu
   int              jaw_shift,        // Jaws kaydırma değeri
   int              teeth_period,     // Teeths (diş) periyodu
   int              teeth_shift,      // Teeths kaydırma değeri
   int              lips_period,      // Lips (dudak) periyodu
   int              lips_shift,       // Lips kaydırma değeri
   ENUM_MA_METHOD   ma_method,        // Ortalama türü
   int              applied           // uygulanacak hacim türü
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

jaw_period

[in] Jaws (çene) periyodu.

jaw_shift

[in] Jaws kaydırma değeri.

teeth_period

[in] Teeths (diş) periyodu.

teeth_shift

[in] Teeths kaydırma değeri.

lips_period

[in] Lips (dudak) periyodu.

lips_shift

[in] Lips kaydırma değeri.

ma_method

[in] Ortalama periyodu (ENUM_MA_METHOD sayımının değerlerinden biri).

applied

[in] Uygulanacak hacim tipi.

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Upper

Üst tamponun belirtilen indisli elemanına dönüş yapar.

```
double  Upper(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Üst tamponun belirtilen indisli elemanına veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

Lower

Alt tamponun belirtilen indisli elemanına dönüş yapar.

```
double  Upper(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Alt tamponun belirtilen indisli elemanına veya geçerli veri yoksa EMPTY_VALUE değerine dönüş yapar.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CiGator için IND_GATOR).

CiBWMFI

CiBWMFI sınıfı, Bill Williams'ın Market Facilitation Index teknik göstergesinin kullanımı için tasarlanmıştır.

Açıklama

CiBWMFI sınıfı, Bill Williams'ın Market Facilitation Index teknik göstergesi için veri erişimi, oluşturma, başlatma gibi uygulamaları içerir.

Bildirim

```
class CiBWMFI: public CIndicator
```

Başlık

```
#include <Indicators\BillWilliams.mqh>
```

Kalıtım hiyerarşisi

CObject
CArray
CArrayObj
CSeries
CIndicator
CiBWMFI

Sınıf Yöntemleri

Özellikler	
Applied	Uygulanacak hacim tipine dönüş yapar
Oluşturma Yöntemleri	
Create	Göstergeyi oluşturur
Veri Erişim Yöntemleri	
Main	Tampon elemanına dönüş yapar
Girdi/çıktı	
virtual Type	Nesne tipi tanımlayıcısına dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler CArray

Step, Step, Total, Available, Max, IsSorted, SortMode, Clear, Sort

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CSeries

Name, BuffersTotal, BufferSize, Timeframe, Symbol, Period, PeriodDescription, RefreshCurrent

Sınıftan türetilen yöntemler CIndicator

Handle, Status, FullRelease, Redrawer, Create, BufferResize, BarsCalculated, GetData, GetData, GetData, GetData, Minimum, MinValue, Maximum, MaxValue, Refresh, AddToChart, DeleteFromChart, MethodDescription, PriceDescription, VolumeDescription

Applied

Uygulanacak hacim tipine dönüş yapar.

```
ENUM_APPLIED_VOLUME  Applied() const
```

Dönüş değeri

Oluşturma sırasında tanımlanan - uygulanacak - hacim tipi.

Create

Belirtilen parametrelerle gösterge oluşturur. Gösterge verilerini güncellemek ve almak için Refresh() ve GetData() yöntemlerini kullanın.

```
bool  Create(
   string               symbol,      // sembol
   ENUM_TIMEFRAMES      period,      // periyot
   ENUM_APPLIED_VOLUME  applied      // uygulanacak hacim
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

applied

[in] Uygulanacak hacim (ENUM_APPLIED_VOLUME sayımının değerlerinden biri).

Dönüş değeri

Başarılı ise 'true', gösterge oluşturulamadıysa 'false'.

Main

Belirtilen indisli tampon elemanına dönüş yapar

```
double  Main(
   int  index      // indis
   )
```

Parametreler

index

[in] Elemanın indisi.

Dönüş değeri

Belirtilen indisli tampon elemanı veya geçerli bir veri yoksa EMPTY_VALUE değeri.

Type

Nesnenin tip tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CiBWMFI için IND_BWMFI).

CiCustom

CiCustom, özel teknik göstergeleri kullanmak için tasarlanmış bir sınıftır.

Açıklama

CiCustom sınıfı, özel göstergeler için oluşturma, başlatma ve veri erişimi gibi yöntemler sağlar.

Bildirim

```
class CiCustom: public CIndicator
```

Başlık

```
#include <Indicators\Custom.mqh>
```

Sınıf Yöntemleri

Özellikler	
NumBuffers	Tampon sayısını ayarlar
NumParams	Parametrelerin sayısını ayarlar
ParamType	Belirtilen parametrenin tipini alır
ParamLong	Belirtilen tamsayı tipli parametrenin değerini alır
ParamDouble	Belirtilen double tipli parametrenin değerini alır
ParamString	Belirtilen string tipli parametrenin değerini alır
Girdi/çıktı	
virtual Type	Gets the object type identifier

NumBuffers

Tampon sayısını ayarlar.

```
bool  NumBuffers(
   int  buffers      // tampon sayısı
   )
```

Dönüş değeri

Başarılı ise 'true', tampon ayarlanmamışsa 'false'.

NumParams

Parametrelerin sayısını alır.

```
int  NumParams() const
```

Dönüş değeri

Gösterge oluşturma sırasında kullanılan parametrelerin sayısı.

ParamType

Belirtilen parametrenin tipini alır.

```
ENUM_DATATYPE  ParamType(
   int  index      // parametre indisi
   ) const
```

Parametreler

index

[in] Parametre indisi.

Dönüş değeri

Gösterge oluşturma sırasında kullanılmış - belirtilen - parametrenin veri tipine dönüş yapar (ENUM_DATATYPE sayımının değerlerinden biri).

Not

Parametre indisi geçersizse WRONG_VALUE dönüşü yapar.

ParamLong

Belirtilen long tipli parametrenin değerini alır.

```
long  ParamLong(
   int  index      // indis
   ) const
```

Parametreler

index

[in] Parametre indisi.

Dönüş değeri

Gösterge oluşturma sırasında kullanılan long tipli parametrenin değeri.

Not

Geçersiz indis değeri girilmişse 0 dönüşü yapar.

ParamDouble

Belirtilen double tipli parametrenin değerini alır.

```
double  ParamDouble(
   int  index      // indis
   ) const
```

Parametreler

index

[in] Parametre indisi.

Dönüş değeri

Gösterge oluşturma sırasında kullanılan double tipli parametrenin değeri.

Not

Geçersiz indis değeri girilmişse EMPTY_VALUE dönüşü yapar.

ParamString

Belirtilen string tipli parametrenin değerini alır.

```
string  ParamString(
   int  index      // indis
   ) const
```

Parametreler

index

[in] Parametre indisi.

Dönüş değeri

Gösterge oluşturma sırasında kullanılan string tipli parametrenin değeri.

Not

Geçersiz indis değeri girilmişse boş bir dizgiye dönüş yapar.

Type

Nesne tipi tanımlayıcısına dönüş yapar.

```
virtual int  Type() const
```

Dönüş değeri

Nesne tipi tanımlayıcısı (CiCustom için IND_CUSTOM).

Alım-Satım Sınıfları

Bu bölüm, alım-satım sınıfları ile çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için gereken açıklamaları içermektedir.

Bu sınıfların kullanımı, özel programlar (Uzman Danışman) oluştururken zaman kazanmanızı sağlayacaktır.

Veri setleri için geliştirilmiş MQL5 Standart Kütüphanesi bileşenleri terminalin çalışma dizininde Include\Arrays klasöründe yer alır.

Sınıf/Grup	Açıklama
CAccountInfo	Alım-satım hesabının özellikleriyle çalışmak için bir sınıf
CSymbolInfo	Alım-satım enstrümanının özellikleriyle çalışmak için bir sınıf
COrderInfo	Bekleyen emirlerin özellikleriyle çalışmak için bir sınıf
CHistoryOrderInfo	Geçmiş emirlerin özellikleriyle çalışmak için bir sınıf
CPositionInfo	Açık pozisyonların özellikleriyle çalışmak için bir sınıf
CDealInfo	Geçmiş işlemlerin özellikleriyle çalışmak için bir sınıf
CTrade	Alım-satım işlemlerinin uygulanması için bir sınıf
CTerminalInfo	Mql5 program ortamı özelliklerinin alınması için bir sınıf

CAccountInfo

CAccountInfo, açık durumdaki işlem hesabı özelliklerine kolay erişim sağlamak için geliştirilmiş bir sınıftır.

Açıklama

CAccountInfo sınıfı, açık durumdaki işlem hesabı özelliklerine kolay erişim sağlar.

Bildirim

```
class CAccountInfo : public CObject
```

Başlık

```
#include <Trade\AccountInfo.mqh>
```

Kalıtım hiyerarşisi

CObject

CAccountInfo

Gruplarına göre sınıf yöntemleri

Tamsayı tipli özelliklere erişim için	
Login	Hesap numarasını alır
TradeMode	Alım-satım modunu alır
TradeModeDescription	İşlem modunu dizgi biçiminde alır
Leverage	Kaldıraç değerini dizgi biçiminde alır
StopoutMode	Hesabın Stop Out modunun değerini alır
StopoutModeDescription	Hesabın Stop Out modunun değerini dizgi biçiminde alır
TradeAllowed	Alım-satım izni bayrağını alır
TradeExpert	Otomatik alım-satım modunun bayrağını alır
LimitOrders	Maksimum bekleyen emir sayısını alır
MarginMode	Teminat hesaplama modunu alır
MarginModeDescription	Teminat hesaplama modunu dizgi biçiminde alır
double tipli özelliklere erişim	
Balance	Hesabın bakiye bilgisini alır
Credit	Verilen kredi miktarını alır
Profit	Hesaptaki mevcut kar miktarını alır

Tamsayı tipli özelliklere erişim için	
Equity	Hesabın mevcut varlık bilgisini alır
Margin	Rezerve edilmiş teminat miktarını alır
FreeMargin	Serbest teminat miktarını alır
MarginLevel	Teminat seviyesini alır
MarginCall	Mevduat için teminat seviyesini alır
MarginStopOut	Stop Out için teminat seviyesini alır
Metin özelliklerine erişim	
Name	Müşteri ismini alır
Server	İşlem sunucusunun ismini alır
Currency	Mevduat cinsinin ismini alır
Company	Hesabın kayıtlı olduğu şirketin ismini alır
MQL5 API fonksiyonlarına erişim	
InfoInteger	Belirtilen tamsayı tipli özelliğin değerini alır
InfoDouble	Belirtilen double tipli özelliğin değerini alır
InfoString	Belirtilen dizgi tipli özelliğin değerini alır.
Ek yöntemler	
OrderProfitCheck	Geçirilen parametrelere göre hesaplanan kar değerini alır
MarginCheck	Alım-satım işlemi gerçekleştirebilmek için gereken teminat miktarını alır
FreeMarginCheck	Alım-satım işleminin uygulanmasının ardından kalan serbest teminat miktarını alır
MaxLotCheck	Alım-satım işlemi için kullanılabilecek maksimum işlem hacmi değerini alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Login

Hesap numarasını alır.

```
long  Login() const
```

Dönüş değeri

Hesap numarası.

TradeMode

Alım-satım modunu alır.

```
ENUM_ACCOUNT_TRADE_MODE  TradeMode() const
```

Dönüş değeri

Alım-satım modu (ENUM_ACCOUNT_TRADE_MODE sayımının değerlerinden biri).

TradeModeDescription

İşlem modunu dizgi biçiminde alır.

```
string  TradeModeDescription() const
```

Dönüş değeri

Dizgi biçiminde, alım-satım modu.

Leverage

Veri kaldıraç miktarını alır.

```
long  Leverage() const
```

Dönüş değeri

Veri kaldıraç miktarı.

StopoutMode

Hesabın Stop Out modunun değerini alır.

```
ENUM_ACCOUNT_STOPOUT_MODE  StopoutMode() const
```

Dönüş değeri

Hesabın Stop Out modu (ENUM_ACCOUNT_STOPOUT_MODE sayımının değerlerinden biri).

StopoutModeDescription

Mevduat için teminat seviyesini alır.

```
string  StopoutModeDescription() const
```

Dönüş değeri

Mevduat için teminat seviyesini alır.

MarginMode

Teminat hesaplama modunu alır.

```
ENUM_ACCOUNT_MARGIN_MODE  MarginMode() const
```

Dönüş Değeri

ENUM_ACCOUNT_MARGIN_MODE sayımının bir değeri şeklinde temiat hesaplama modu.

MarginModeDescription

Teminat hesaplama modunu dizgi biçiminde alır.

```
string  MarginModeDescription() const
```

Dönüş Değeri

Dizgi biçiminde teminat hesaplama modu.

TradeAllowed

Alım-satım izni bayrağını alır.

```
bool  TradeAllowed() const
```

Dönüş değeri

Alım-satım izni bayrağı.

TradeExpert

Otomatik alım-satım izninin bayrağını alır.

```
bool  TradeExpert() const
```

Dönüş değeri

Otomatik alım-satım izni bayrağı.

LimitOrders

Maksimum bekleyen emir sayısını alır

```
int  LimitOrders() const
```

Dönüş değeri

Maksimum bekleyen emir sayısı.

Not

0 - limit yok.

Balance

Hesabın bakiye bilgisini alır.

```
double  Balance() const
```

Dönüş değeri

Hesabın bakiye bilgisi (mevduat birimi cinsinden).

Credit

Verilen kredi miktarını alır.

```
double  Credit() const
```

Dönüş değeri

Verilen kredi miktarı (mevduat birimi cinsinden).

Profit

Hesaptaki mevcut kar miktarını alır.

```
double  Profit() const
```

Dönüş değeri

Hesaptaki mevcut kar miktarını - mevduat dövizinin cinsinden - alır.

Equity

Hesabın mevcut varlık bilgisini alır

```
double  Equity() const
```

Dönüş değeri

Hesabın mevcut öz-kaynak bilgisi (mevduat birimi cinsinden).

Margin

Rezerve edilmiş teminat miktarını alır.

```
double  Margin() const
```

Dönüş değeri

Rezerve edilmiş teminat miktarı (mevduat birimi cinsinden).

FreeMargin

Serbest teminat miktarını alır.

```
double  FreeMargin() const
```

Dönüş değeri

Serbest teminat miktarı (mevduat birimi cinsinden).

MarginLevel

Teminat seviyesini alır.

```
double  MarginLevel() const
```

Dönüş değeri

Teminat seviyesi.

MarginCall

Mevduat için teminat seviyesini alır.

```
double  MarginCall() const
```

Dönüş değeri

Mevduat için teminat seviyesi.

MarginStopOut

Stop Out için teminat seviyesini alır.

```
double  MarginStopOut() const
```

Dönüş değeri

Stop Out olayı için teminat seviyesi.

Name

Müşteri ismini alır.

```
string  Name() const
```

Dönüş değeri

Müşteri ismi.

Server

İşlem sunucusunun ismini alır.

```
string  Server() const
```

Dönüş değeri

İşlem sunucusunun ismi.

Currency

Mevduat cinsinin ismini alır.

```
string  Currency() const
```

Dönüş değeri

Mevduat para birimi.

Company

Hesabın kayıtlı olduğu şirketin ismini alır

```
string  Company() const
```

Dönüş değeri

Hesabın kayıtlı olduğu şirketin ismi

InfoInteger

Belirtilen tamsayı tipli özelliğin değerini alır.

```
long  InfoInteger(
   ENUM_ACCOUNT_INFO_INTEGER  prop_id     // özellik tanımlayıcı
   ) const
```

Parametreler

prop_id

[in] Özellik tanımlayıcı. ENUM_ACCOUNT_INFO_INTEGER sayımının değerlerinden biri olabilir.

Dönüş değeri

long tipli değer.

InfoDouble

Belirtilen double tipli özelliğin değerini alır.

```
double  InfoDouble(
   ENUM_ACCOUNT_INFO_DOUBLE  prop_id     // özellik tanımlayıcı
   ) const
```

Parametreler

prop_id

[in] Özellik tanımlayıcı. ENUM_ACCOUNT_INFO_DOUBLE sayımının değerlerinden biri olabilir.

Dönüş değeri

double tipli değer.

InfoString

Belirtilen string tipli özelliğin değerini alır.

```
string  InfoString(
   ENUM_ACCOUNT_INFO_STRING  prop_id     // özellik tanımlayıcısı
   ) const
```

Parametreler

prop_id

[in] Özellik tanımlayıcı. ENUM_ACCOUNT_INFO_STRING sayımının değerlerinden biri olabilir.

Dönüş değeri

string tipli değer.

OrderProfitCheck

Bu fonksiyon, geçirilen parametrelere göre mevcut hesap için kar değerini hesaplar. Alım-satım işleminin ön-değerlendirmesi için kullanılır. Değere mevduat dövizinin cinsinden dönüş yapar.

```
double  OrderProfitCheck(
   const string        symbol,              // sembol
   ENUM_ORDER_TYPE     trade_operation,     // işlem tipi (ORDER_TYPE_BUY veya ORDER_T
   double              volume,              // hacim
   double              price_open,          // açılış fiyatı
   double              price_close          // kapanış fiyatı
   ) const
```

Parametreler

symbol

[in] Alım-satım işlemi için kullanılacak sembol.

trade_operation

[in] Alım-satım işleminin tipi (ENUM_ORDER_TYPE sayımının değerlerinden biri).

volume

[in] Alım-satım işleminin hacmi.

price_open

[in] Açılış fiyatı.

price_close

[in] Kapanış fiyatı.

Dönüş değeri

Başarı durumunda kar miktarına, aksi durumda EMPTY_VALUE değerine dönüş yapar.

MarginCheck

Alım-satım işlemi gerçekleştirebilmek için gereken teminat miktarını alır.

```
double  MarginCheck(
   const string         symbol,              // sembol
   ENUM_ORDER_TYPE      trade_operation,     // işlem
   double               volume,              // hacim
   double               price                // fiyat
   ) const
```

Parametreler

symbol

[in] Alım-satım işlemi için kullanılacak sembol.

trade_operation

[in] (ENUM_ORDER_TYPE sayımı).

volume

[in] Alım-satım işleminin hacmi.

price

[in] Alım-satım işleminin fiyatı.

Dönüş değeri

Alım-satım işlemi gerçekleştirebilmek için gereken teminat miktarı.

FreeMarginCheck

Alım-satım işleminin uygulanmasının ardından kalan serbest teminat miktarını alır.

```
double  FreeMarginCheck(
   const string         symbol,              // sembol
   ENUM_ORDER_TYPE      trade_operation,     // işlem
   double               volume,              // hacim
   double               price                // fiyat
   ) const
```

Parametreler

symbol

[in] Alım-satım işlemi için kullanılacak sembol.

trade_operation

[in] (ENUM_ORDER_TYPE sayımı).

volume

[in] Alım-satım işleminin hacmi.

price

[in] Alım-satım işleminin fiyatı.

Dönüş değeri

Alım-satım işleminin uygulanmasının ardından kalan serbest teminat miktarı.

MaxLotCheck

Alım-satım işlemi için kullanılabilecek maksimum işlem hacmi değerini alır.

```
double  MaxLotCheck(
   const string         symbol,              // sembol
   ENUM_ORDER_TYPE      trade_operation,     // işlem
   double               price,               // fiyat
   double               percent=100          // alım-satım için kullanılabilir mevcut t
   ) const
```

Parametreler

symbol

[in] Alım-satım işlemi için kullanılacak sembol.

trade_operation

[in] Alım-satım işleminin tipi (ENUM_ORDER_TYPE sayımının değeri).

price

[in] Alım-satım işleminin fiyatı.

percent=100

[in] Alım-satım için kullanılabilir teminat yüzdesi.

Dönüş değeri

Alım-satım işleminin maksimum hacmi.

CSymbolInfo

CSymbolInfo sınıfı, sembol özelliklerine kolay erişim için tasarlanmıştır.

Açıklama

CSymbolInfo sınıfı, sembol özelliklerine erişim sağlar.

Bildirim

```
class CSymbolInfo : public CObject
```

Başlık

```
#include <Trade\SymbolInfo.mqh>
```

Kalıtım hiyerarşisi

CObject

CSymbolInfo

Gruplarına göre sınıf yöntemleri

Kontrol	
Refresh	Sembol verisini yeniler
RefreshRates	Sembolün kotasyon verilerini yeniler
Özellikler	
Name	Sembol ismini alır/ayarlar
Select	"Piyasa Gözlemi" sembolünün bayrağını alır/ayarlar
IsSynchronized	Sembolün sunucu verileriyle olan senkronizasyonunu denetler
Hacimler	
Volume	Son işlemin hacmini alır
VolumeHigh	Günün maksimal hacim değerini alır
VolumeLow	Günün minimal hacim değerini alır
Çeşitli	
Time	Son kotasyon zamanını alır
Spread	Spread miktarını puan cinsinden alır
SpreadFloat	Dalgalı (değişken) spread bayrağını alır
TicksBookDepth	Tik depolama derinliğini alır
Seviyeler	

Kontrol	
StopsLevel	Emirler için puan bazında minimal sipariş değerini alır
FreezeLevel	Alım-satım işlemlerini dondurma noktasının uzaklığını puan bazında alır
Bid (teklif) fiyatları	
Bid	Mevcut Bid fiyatını alır
BidHigh	Günün maksimal Bid fiyatını alır
BidLow	Günün minimal Bid fiyatını alır
Ask (istek) fiyatları	
Ask	Mevcut Ask fiyatını alır
AskHigh	Günün maksimal Ask fiyatını alır
AskLow	Günün minimal Ask fiyatını alır
Fiyatlar	
Last	Mevcut Last (son) fiyatını alır
LastHigh	Bir gün için maksimal Last fiyatını alır
LastLow	Bir gün için minimal Last fiyatını alır
Alım-satım modları	
TradeCalcMode	Kontrat maliyeti hesabının modunu alır
TradeCalcModeDescription	Kontrat maliyet hesabı modunu dizgi biçiminde alır
TradeMode	Emir uygulama türünü alır
TradeModeDescription	Emir uygulama türünü dizgi biçiminde alır
TradeExecution	İşlem kapama modunu alır
TradeExecutionDescription	İşlem kapama modunu dizgi biçiminde alır
Swaplar	
SwapMode	Swap hesaplama modunu alır
SwapModeDescription	Swap hesaplama modelini dizgi biçiminde alır
SwapRollover3days	Üçlü swap ücreti gününü alır
SwapRollover3daysDescription	Üçlü swap ücreti gününü dizgi biçiminde alır
Teminatlar ve bayraklar	
MarginInitial	Başlangıç teminat değerini alır
MarginMaintenance	İdame teminatının değerini alır
MarginLong	Uzun pozisyonlar için teminat bedeli oranını alır

Kontrol	
MarginShort	Kısa pozisyonlar için teminat bedeli oranını alır
MarginLimit	Limit emirleri için teminat bedeli oranını alır
MarginStop	Stop emirleri için teminat bedeli oranını alır
MarginStopLimit	StopLimit emirleri için teminat bedeli oranını alır
TradeTimeFlags	İzin verilen emir zaman-aşımı modlarının bayraklarını alır
TradeFillFlags	İzin verilen emir karşılama modlarının bayraklarını alır
Niceleme	
Digits	Sembolün ondalık basamaklarının sayısını alır
Point	Puan değerini alır
TickValue	Tik (minimal fiyat değişimi) maliyetini alır
TickValueProfit	Karlı pozisyon için hesaplanan tik fiyatını alır
TickValueLoss	Zarar eden pozisyon için hesaplanan tik fiyatını alır
TickSize	Minimal fiyat değişimi değerini alır
Kontrat büyüklüklükleri	
ContractSize	İşlem kontratının meblağını alır
LotsMin	Bir işlemi kapamak için gereken minimal hacim değerini alır
LotsMax	Bir işlemi kapamak için gereken maksimal hacim değerini alır
LotsStep	Bir işlemi kapamak için gereken minimal hacim değişim adımının değerini alır
LotsLimit	Bir sembol üzerinde açık pozisyonlar ve bekleyen emirler için izin verilen maksimum hacim değerini alır
Swap değerleri	
SwapLong	Uzun pozisyon için swap değerini alır
SwapShort	Kısa pozisyon için swap değerini alır
Metin özellikleri	
CurrencyBase	Baz dövizin sembol ismini alır
CurrencyProfit	Kar dövizinin ismini alır
CurrencyMargin	Teminat dövizinin ismini alır
Bank	Mevcut kotasyonun kaynağının ismini alır
Açıklama	Sembolün dizgi şeklindeki açıklamasını alır
Path	Sembol ağacındaki adresi alır

Kontrol	
Sembol özellikleri	
SessionDeals	Mevcut seanstaki işlemlerin sayısını alır
SessionBuyOrders	Mevcut Alış emirlerinin sayısını alır
SessionSellOrders	Mevcut Satış emirlerinin sayısını alır
SessionTurnover	Mevcut seans cirosunun özetini alır
SessionInterest	Mevcut seansın açılış faizini alır
SessionBuyOrdersVolume	Alış emirlerinin mevcut hacmini alır
SessionSellOrdersVolume	Satış emirlerinin mevcut hacmini alır
SessionOpen	Mevcut seansın açılış fiyatını alır
SessionClose	Mevcut seansın kapanış fiyatını alır
SessionAW	Mevcut seansın ağırlıklı ortalama fiyatını alır
SessionPriceSettlement	Mevcut seansın uzlaşma fiyatını alır
SessionPriceLimitMin	Mevcut seansın minimal fiyatını alır
SessionPriceLimitMax	Mevcut seansın maksimal fiyatını alır
MQL5 API fonksiyonlarına erişim	
InfoInteger	Belirtilen tamsayı tipli özelliğin değerini alır
InfoDouble	Belirtilen double tipli özelliğin değerini alır
InfoString	Belirtilen string tipli özelliğin değerini alır.
Hizmet fonksiyonları	
NormalizePrice	Sembol özellikleri kullanılarak normalleştirilmiş olan fiyat değerine dönüş yapar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Refresh

Sembol verisini yeniler.

```
void  Refresh()
```

Dönüş değeri

Yok.

Not

Sembol, Name yöntemiyle seçilmelidir.

RefreshRates

Sembolün kotasyon verilerini yeniler.

```
bool  RefreshRates()
```

Dönüş değeri

Başarılı ise 'true', kotasyon verilerini yenilenemezse 'false'.

Not

Sembol, Name yöntemiyle seçilmelidir.

Name

Sembol ismini alır.

```
string  Name() const
```

Dönüş değeri

Sembol ismi.

Name

Sembol ismini ayarlar.

```
bool  Name(string name)
```

Dönüş değeri

Yok.

Select

"Piyasa Gözlemi" sembolünün bayrağını alır.

```
bool  Select() const
```

Dönüş değeri

"Piyasa Gözlemi" sembolünün bayrağını alır.

Select

"Piyasa Gözlemi" sembolünün bayrağını ayarlar.

```
bool  Select()
```

Dönüş değeri

Başarılı ise 'true', bayrak değiştirilemezse 'false'.

IsSynchronized

Sembolün sunucu verileriyle olan senkronizasyonunu denetler.

```
bool  IsSynchronized() const
```

Dönüş değeri

true - sembol sunucu ile senkronize ise; false - değilse.

Not

Sembol, Name yöntemiyle seçilmelidir.

Volume

Son işlemin hacmini alır.

```
long  Volume() const
```

Dönüş değeri

Son işlemin hacmi.

Not

Sembol, Name yöntemiyle seçilmelidir.

VolumeHigh

Günün maksimal hacim değerini alır

```
long  VolumeHigh() const
```

Dönüş değeri

Günün maksimal hacim değeri.

Not

Sembol, Name yöntemiyle seçilmelidir.

VolumeLow

Günün minimal hacim değerini alır.

```
long  VolumeLow() const
```

Dönüş değeri

Günün minimal hacim değeri.

Not

Sembol, Name yöntemiyle seçilmelidir.

Time

Son kotasyon zamanını alır.

```
datetime  Time() const
```

Dönüş değeri

Son kotasyon zamanı.

Not

Sembol, Name yöntemiyle seçilmelidir.

Spread

Puan cinsinden spread miktarı

```
int  Spread() const
```

Dönüş değeri

Puan cinsinden spread miktarı

Not

Sembol, Name yöntemiyle seçilmelidir.

SpreadFloat

Dalgalı (değişken) spread bayrağını alır.

```
bool  SpreadFloat() const
```

Dönüş değeri

Dalgalı spread bayrağı.

Not

Sembol, Name yöntemiyle seçilmelidir.

TicksBookDepth

Tik depolama derinliğini alır.

```
int  TicksBookDepth() const
```

Dönüş değeri

Tik depolama derinliği.

Not

Sembol, Name yöntemiyle seçilmelidir.

StopsLevel

Emirler için puan bazında minimal stop seviyesini alır.

```
int  StopsLevel() const
```

Dönüş değeri

Emirler için puan bazında minimal stop seviyesi.

Not

Sembol, Name yöntemiyle seçilmelidir.

FreezeLevel

İşlem dondurma seviyesini puan bazında alır.

```
int  FreezeLevel() const
```

Dönüş değeri

İşlem dondurma seviyesi (puan bazında).

Not

Sembol, Name yöntemiyle seçilmelidir.

Bid

Mevcut satış fiyatını alır.

```
double  Bid() const
```

Dönüş değeri

Mevcut Bid (teklif) fiyatı.

Not

Sembol, Name yöntemiyle seçilmelidir.

BidHigh

Günün maksimal Bid (teklif) fiyatını alır

```
double  BidHigh() const
```

Dönüş değeri

Günün maksimal Bid fiyatı.

Not

Sembol, Name yöntemiyle seçilmelidir.

BidLow

Günün minimal Bid (teklif) fiyatını alır

```
double  BidLow() const
```

Dönüş değeri

Günün minimal Bid fiyatı.

Not

Sembol, Name yöntemiyle seçilmelidir.

Ask

Mevcut alış fiyatını alır.

```
double  Ask() const
```

Dönüş değeri

Mevcut alış fiyatı.

Not

Sembol, Name yöntemiyle seçilmelidir.

AskHigh

Bir gün için maksimal alış fiyatını alır.

```
double  AskHigh() const
```

Dönüş değeri

Bir gün için maksimal alış fiyatı.

Not

Sembol, Name yöntemiyle seçilmelidir.

AskLow

Bir gün için minimal Ask (istek) fiyatını alır.

```
double  AskLow() const
```

Dönüş değeri

Bir gün için minimal Ask fiyatı.

Not

Sembol, Name yöntemiyle seçilmelidir.

Last

Mevcut Last (son) fiyatı alır.

```
double  Last() const
```

Dönüş değeri

Mevcut Last fiyatı.

LastHigh

Günün maksimal Last (son) fiyatını alır

```
double  LastHigh() const
```

Dönüş değeri

Günün maksimal Last fiyatı.

Not

Sembol, Name yöntemiyle seçilmelidir.

LastLow

Günün minimal Last (son) fiyatını alır

```
double  LastLow() const
```

Dönüş değeri

Günün minimal Last fiyatı.

Not

Sembol, Name yöntemiyle seçilmelidir.

TradeCalcMode

Kontrat maliyeti hesabının modunu alır.

```
ENUM_SYMBOL_CALC_MODE  TradeCalcMode() const
```

Dönüş değeri

Kontrat maliyeti hesabının modu (ENUM_SYMBOL_CALC_MODE sayımının değerlerinden biri).

Not

Sembol, Name yöntemiyle seçilmelidir.

TradeCalcModeDescription

Kontrat maliyet hesabı modunu dizgi biçiminde alır.

```
string  TradeCalcModeDescription() const
```

Dönüş değeri

Dizgi biçiminde kontrat maliyet hesabı modu.

Not

Sembol, Name yöntemiyle seçilmelidir.

TradeMode

Emir uygulama tipini alır.

```
ENUM_SYMBOL_TRADE_MODE  TradeMode() const
```

Dönüş değeri

Emir uygulama tipi (ENUM_SYMBOL_TRADE_MODE sayımının değerlerinden biri).

Not

Sembol, Name yöntemiyle seçilmelidir.

TradeModeDescription

İşlem modunu dizgi biçiminde alır.

```
string  TradeModeDescription() const
```

Dönüş değeri

Dizgi biçiminde, alım-satım modu.

Not

Sembol, Name yöntemiyle seçilmelidir.

TradeExecution

Alım-satım uygulama modunu alır.

```
ENUM_SYMBOL_TRADE_EXECUTION  TradeExecution() const
```

Dönüş değeri

Alım-satım uygulama modu (ENUM_SYMBOL_TRADE_EXECUTION sayımının değerlerinden biri).

Not

Sembol, Name yöntemiyle seçilmelidir.

TradeExecutionDescription

Alım-satım uygulama modunun açıklamasını dizgi biçiminde alır.

```
string  TradeExecutionDescription() const
```

Dönüş değeri

Dizgi biçiminde, alım-satım uygulama modu.

Not

Sembol, Name yöntemiyle seçilmelidir.

SwapMode

Swap hesaplama modunu alır.

```
ENUM_SYMBOL_SWAP_MODE  SwapMode() const
```

Dönüş değeri

Swap hesaplama modu (ENUM_SYMBOL_SWAP_MODE sayımının değerlerinden biri).

Not

Sembol, Name yöntemiyle seçilmelidir.

SwapModeDescription

Swap hesaplama modunu dizgi biçiminde alır

```
string  SwapModeDescription() const
```

Dönüş değeri

Dizgi biçiminde swap hesaplama modu.

Not

Sembol, Name yöntemiyle seçilmelidir.

SwapRollover3days

Üçlü swap ücreti gününü alır.

```
ENUM_DAY_OF_WEEK  SwapRollover3days() const
```

Dönüş değeri

Üçlü swap ücreti gününü (ENUM_DAY_OF_WEEK sayımının değerlerinden biri).

Not

Sembol, Name yöntemiyle seçilmelidir.

SwapRollover3daysDescription

Üçlü swap ücreti gününü dizgi biçiminde alır.

```
string  SwapRollover3daysDescription() const
```

Dönüş değeri

Dizgi biçiminde üçlü swap ücreti günü.

Not

Sembol, Name yöntemiyle seçilmelidir.

MarginInitial

Başlangıç teminat değerini alır.

```
double  MarginInitial()
```

Dönüş değeri

Başlangıç teminat değeri.

Not

Lot değerinden ücretlendirilen teminat (sembolün teminat dövizi cinsinden) miktarına işaret eder. Piyasaya giren müşterinin sermayesini kontrol etmek için kullanılır.

Sembol, Name yöntemiyle seçilmelidir.

MarginMaintenance

İdame teminatının değerini alır.

```
double  MarginMaintenance()
```

Dönüş değeri

İdame teminatının değeri.

Not

Lot değerinden ücretlendirilen teminat (sembolün teminat dövizi cinsinden) miktarına işaret eder. Hesap durumu değiştiğinde müşterinin sermayesini kontrol etmek için kullanılır. İdame teminatı 0 ise, başlangıç teminatı kullanılır.

Sembol, Name yöntemiyle seçilmelidir.

MarginLong

Uzun pozisyonlar için teminat bedeli oranını alır

```
double  MarginLong() const
```

Dönüş değeri

Uzun pozisyonlar için teminat bedeli oranı.

Not

Sembol, Name yöntemiyle seçilmelidir.

MarginShort

Gets the rate of margin charging on short positons.

```
double  MarginShort() const
```

Dönüş değeri

Rate of margin charging on short positons.

Not

Sembol, Name yöntemiyle seçilmelidir.

MarginLimit

Limit emirleri için teminat bedeli oranını alır.

```
double  MarginLimit() const
```

Dönüş değeri

Limit emirleri için teminat bedeli oranı.

Not

Sembol, Name yöntemiyle seçilmelidir.

MarginStop

Stop emirleri için teminat bedeli oranını alır.

```
double  MarginStop() const
```

Dönüş değeri

Stop emirleri için teminat bedeli oranı.

Not

Sembol, Name yöntemiyle seçilmelidir.

MarginStopLimit

Stop Limit emirleri için teminat bedeli oranını alır.

```
double  MarginStopLimit() const
```

Dönüş değeri

Stop Limit emirleri için teminat bedeli oranı.

Not

Sembol, Name yöntemiyle seçilmelidir.

TradeTimeFlags

İzin verilen emir zaman-aşımı modlarının bayraklarını alır.

```
int  TradeTimeFlags() const
```

Dönüş değeri

İzin verilen emir zaman-aşımı modlarının bayrakları.

Not

Sembol, Name yöntemiyle seçilmelidir.

TradeFillFlags

İzin verilen emir karşılama modlarının (emir türleri) bayraklarını alır.

```
int  TradeFillFlags() const
```

Dönüş değeri

İzin verilen emir karşılama modlarının bayrakları.

Not

Sembol, Name yöntemiyle seçilmelidir.

Digits

Sembolün ondalık basamaklarının sayısı.

```
int  Digits() const
```

Dönüş değeri

Sembolün ondalık basamaklarının sayısı.

Not

Sembol, Name yöntemiyle seçilmelidir.

Point

Puan değerini alır.

```
double  Point() const
```

Dönüş değeri

Puan değeri.

Not

Sembol, Name yöntemiyle seçilmelidir.

TickValue

Tik (minimal fiyat değişimi) maliyetini alır.

```
double  TickValue() const
```

Dönüş değeri

Tik (minimal fiyat değişimi) maliyeti.

Not

Sembol, Name yöntemiyle seçilmelidir.

TickValueProfit

Karlı pozisyon için hesaplanan tik fiyatını alır.

```
double  TickValueProfit() const
```

Dönüş değeri

Karlı pozisyon için hesaplanan tik fiyatı.

Not

Sembol, Name yöntemiyle seçilmelidir.

TickValueLoss

Zarar eden pozisyon için hesaplanan tik fiyatını alır.

```
double  TickValueLoss() const
```

Dönüş değeri

Zarar eden pozisyon için hesaplanan tik fiyatı.

Not

Sembol, Name yöntemiyle seçilmelidir.

TickSize

Minimal fiyat değişimi değerini alır.

```
double  TickSize() const
```

Dönüş değeri

Minimal fiyat değişimi.

Not

Sembol, Name yöntemiyle seçilmelidir.

ContractSize

İşlem kontratının meblağını alır.

```
double  ContractSize() const
```

Dönüş değeri

İşlem kontratının meblağı.

Not

Sembol, Name yöntemiyle seçilmelidir.

LotsMin

Bir işlemi kapamak için gereken minimal hacim değerini alır.

```
double  LotsMin() const
```

Dönüş değeri

Bir işlemi kapamak için gereken minimal hacim değeri.

Not

Sembol, Name yöntemiyle seçilmelidir.

LotsMax

Bir işlemi kapamak için gereken maksimal hacim değerini alır.

```
double  LotsMax() const
```

Dönüş değeri

Bir işlemi kapamak için gereken maksimal hacim değeri.

Not

Sembol, Name yöntemiyle seçilmelidir.

LotsStep

Bir işlemi kapamak için gereken minimal hacim değişim adımının değerini alır.

```
double  LotsStep() const
```

Dönüş değeri

Bir işlemi kapamak için gereken minimal hacim değişim adımının değeri.

Not

Sembol, Name yöntemiyle seçilmelidir.

LotsLimit

Bir sembol üzerinde (yön bağımsız) açık pozisyonlar ve bekleyen emirler için izin verilen maksimum hacim değerini alır

```
double  LotsLimit() const
```

Dönüş değeri

Bir sembol üzerinde (yön bağımsız) açık pozisyonlar ve bekleyen emirler için izin verilen maksimum hacim değeri.

Not

Sembol, Name yöntemiyle seçilmelidir.

SwapLong

Uzun pozisyon için swap değerini alır.

```
double  SwapLong() const
```

Dönüş değeri

Uzun pozisyon için swap değeri.

Not

Sembol, Name yöntemiyle seçilmelidir.

SwapShort

Kısa pozisyon için swap değerini alır.

```
double  SwapShort() const
```

Dönüş değeri

Kısa pozisyon için swap değeri.

Not

Sembol, Name yöntemiyle seçilmelidir.

CurrencyBase

Baz dövizin sembol ismini alır.

```
string  CurrencyBase() const
```

Dönüş değeri

Baz dövizin sembol ismi.

Not

Sembol, Name yöntemiyle seçilmelidir.

CurrencyProfit

Kar dövizinin ismini alır.

```
string  CurrencyProfit() const
```

Dönüş değeri

Kar dövizinin ismi.

Not

Sembol, Name yöntemiyle seçilmelidir.

CurrencyMargin

Teminat dövizinin ismini alır.

```
string  CurrencyMargin() const
```

Dönüş değeri

Teminat dövizinin ismi.

Not

Sembol, Name yöntemiyle seçilmelidir.

Bank

Mevcut kotasyonun kaynağının ismini alır.

```
string  Bank() const
```

Dönüş değeri

Mevcut kotasyonun kaynağının ismi.

Not

Sembol, Name yöntemiyle seçilmelidir.

Açıklama

Sembolün dizgi şeklindeki açıklamasını alır.

```
string  Description() const
```

Dönüş değeri

Sembolün dizgi şeklindeki açıklaması.

Not

Sembol, Name yöntemiyle seçilmelidir.

Path

Sembol ağacındaki adresi alır.

```
string  Path() const
```

Dönüş değeri

Sembol ağacındaki adresi alır.

Not

Sembol, Name yöntemiyle seçilmelidir.

SessionDeals

Mevcut seanstaki işlemlerin sayısını alır.

```
long  SessionDeals() const
```

Dönüş değeri

Mevcut seanstaki işlemlerin sayısı.

Not

Sembol, Name yöntemiyle seçilmelidir.

SessionBuyOrders

Mevcut Alış emirlerinin sayısını alır.

```
long  SessionBuyOrders() const
```

Dönüş değeri

Mevcut Alış emirlerinin sayısı.

Not

Sembol, Name yöntemiyle seçilmelidir.

SessionSellOrders

Mevcut Satış emirlerinin sayısını alır.

```
long  SessionSellOrders() const
```

Dönüş değeri

Mevcut Satış emirlerinin sayısı.

Not

Sembol, Name yöntemiyle seçilmelidir.

SessionTurnover

Mevcut seans cirosunun özetini alır.

```
double  SessionTurnover() const
```

Dönüş değeri

Mevcut seans cirosunun özeti.

Not

Sembol, Name yöntemiyle seçilmelidir.

SessionInterest

Mevcut seansın açılış faizini alır.

```
double  SessionInterest() const
```

Dönüş değeri

Mevcut seansın açılış faizi.

Not

Sembol, Name yöntemiyle seçilmelidir.

SessionBuyOrdersVolume

Alış emirlerinin mevcut hacmini alır.

```
double  SessionBuyOrdersVolume() const
```

Dönüş değeri

Alış emirlerinin mevcut hacmi.

Not

Sembol, Name yöntemiyle seçilmelidir.

SessionSellOrdersVolume

Satış emirlerinin mevcut hacmini alır.

```
double  SessionSellOrdersVolume() const
```

Dönüş değeri

Satış emirlerinin mevcut hacmi.

Not

Sembol, Name yöntemiyle seçilmelidir.

SessionOpen

Mevcut seansın açılış fiyatını alır.

```
double  SessionOpen() const
```

Dönüş değeri

Mevcut seansın açılış fiyatı.

Not

Sembol, Name yöntemiyle seçilmelidir.

SessionClose

Mevcut seansın kapanış fiyatını alır.

```
double  SessionClose() const
```

Dönüş değeri

Mevcut seansın kapanış fiyatı.

Not

Sembol, Name yöntemiyle seçilmelidir.

SessionAW

Mevcut seansın ağırlıklı ortalama fiyatını alır.

```
double  SessionAW() const
```

Dönüş değeri

Mevcut seansın ağırlıklı ortalama fiyatı.

Not

Sembol, Name yöntemiyle seçilmelidir.

SessionPriceSettlement

Mevcut seansın uzlaşma fiyatını alır.

```
double  SessionPriceSettlement() const
```

Dönüş değeri

Mevcut seansın uzlaşma fiyatı.

Not

Sembol, Name yöntemiyle seçilmelidir.

SessionPriceLimitMin

Mevcut seansın minimal fiyatını alır.

```
double  SessionPriceLimitMin() const
```

Dönüş değeri

Mevcut seansın minimal fiyatı.

Not

Sembol, Name yöntemiyle seçilmelidir.

SessionPriceLimitMax

Mevcut seansın maksimal fiyatını alır.

```
double  SessionPriceLimitMax() const
```

Dönüş değeri

Mevcut seansın maksimal fiyatı.

Not

Sembol, Name yöntemiyle seçilmelidir.

InfoInteger

Belirtilen tamsayı tipli özelliğin değerini alır.

```
bool  InfoInteger(
   ENUM_SYMBOL_INFO_INTEGER  prop_id,      // özellik tanımlayıcı
   long&                     var           // değişken referansı
   ) const
```

Parametreler

prop_id

[in] Tamsayı tipli özelliğin tanımlayıcısı (ENUM_SYMBOL_INFO_INTEGER sayımının değerlerinden biri).

var

[out] Sonucu yerleştirmek için kullanılacak long tipli değişkenin referansı.

Dönüş değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Not

Sembol, Name yöntemiyle seçilmelidir.

InfoDouble

Belirtilen double tipli özelliğin değerini alır.

```
bool  InfoDouble(
   ENUM_SYMBOL_INFO_DOUBLE  prop_id,      // özellik tanımlayıcı
   double&                  var           // değişken referansı
   ) const
```

Parametreler

prop_id

[in] double tipli özelliğin tanımlayıcısı (ENUM_SYMBOL_INFO_DOUBLE sayımının değerlerinden biri).

var

[out] Sonucu yerleştirmek için kullanılacak olan double tipli değişkenin referansı.

Dönüş değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Not

Sembol, Name yöntemiyle seçilmelidir.

InfoString

Belirtilen string tipli özelliğin değerini alır.

```
bool  InfoString(
   ENUM_SYMBOL_INFO_STRING  prop_id,      // özellik tanımlayıcı
   string&                  var           // değişken referansı
   ) const
```

Parametreler

prop_id

[in] Dizgi biçimli özelliğin tanımlayıcısı.

var

[out] Sonucu yerleştirmek için kullanılacak string tipli değişkenin referansı.

Dönüş değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Not

Sembol, Name yöntemiyle seçilmelidir.

NormalizePrice

Sembol özellikleri kullanılarak normalleştirilmiş olan fiyat değerine dönüş yapar.

```
double  NormalizePrice(
   double  price   // fiyat
   ) const
```

Parametreler

price

[in] Fiyat.

Dönüş değeri

Normalleştirilmiş fiyat.

Not

Sembol, Name yöntemiyle seçilmelidir.

COrderInfo

COrderInfo sınıfı, bekleyen emirlerin özelliklerine kolay erişim için tasarlanmıştır.

Açıklama

COrderInfo sınıfı, bekleyen emirlerin özelliklerine erişim sağlar.

Bildirim

```
class COrderInfo : public CObject
```

Başlık

```
#include <Trade\OrderInfo.mqh>
```

Kalıtım hiyerarşisi

CObject

COrderInfo

Gruplarına göre sınıf yöntemleri

Tamsayı tipli özelliklere erişim için	
Ticket	Erişim için önceden seçilmiş olan emrin fişini alır
TimeSetup	Emir yerleştirme zamanını alır
TimeSetupMsc	Emir yerleştirme zamanını 01.01.1970 tarihinden itibaren milisaniyeler cinsinden alır
OrderType	Emir tipini alır
OrderTypeDescription	Emir tipini dizgi biçiminde alır
State	Emir durumunu alır
StateDescription	Emir durumunu dizgi biçiminde alır
TimeExpiration	Emir zaman-aşımı değerini alır
TimeDone	Emir uygulama veya iptal zamanını alır
TimeDoneMsc	Emir uygulama veya iptal zamanını 01.01.1970 tarihinden itibaren milisaniyeler cinsinden alır
TypeFilling	Emir uygulama türünü alır
TypeFillingDescription	Emir uygulama türünü dizgi biçiminde alır
TypeTime	Zaman-aşımı anında emir tipini alır
TypeTimeDescription	Zaman-aşımı anında emir tipini dizgi şeklinde alır
Magic	Emri yerleştiren Uzmanın tanımlayıcısını alır

Tamsayı tipli özelliklere erişim için	
PositionId	Pozisyonun tanımlayıcısını alır
double tipli özelliklere erişim	
VolumeInitial	Emrin başlangıç hacim değerini alır
VolumeCurrent	Emrin karşılanmamış hacim değerini alır
PriceOpen	Emir fiyatını alır
StopLoss	Emrin Stop Loss fiyatını alır
TakeProfit	Emrin Take Profit fiyatını alır
PriceCurrent	Emir sembolünün mevcut fiyatını alır
PriceStopLimit	Ayarlanan limit emrinin fiyatını alır
Metin özelliklerine erişim	
Symbol	Emir sembolünün ismini alır
Comment	Emir yorumunu alır
MQL5 API fonksiyonlarına erişim	
InfoInteger	Belirtilen tamsayı tipli özelliğin değerini alır
InfoDouble	Belirtilen double tipli özelliğin değerini alır
InfoString	Belirtilen string tipli özelliğin değerini alır
Durum	
StoreState	Emir parametrelerini kaydeder
CheckState	Mevcut parametreleri kaydedilen parametrelerle karşılaştırır
Seçim	
Select	Özelliklerine erişebilmek için fiş kullanarak bir emir seçer.
SelectByIndex	Özelliklerine erişebilmek için indis kullanarak bir emir seçer.

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Ticket

Select yöntemini kullanarak önceden, erişim seçilmiş olan emrin fişini alır.

```
ulong  Ticket() const
```

Dönüş değeri

Başarılıysa emir fişine, aksi durumda - ULONG_MAX değerine dönüş yapar.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

TimeSetup

Emir yerleştirme zamanını alır.

```
datetime  TimeSetup() const
```

Dönüş değeri

Emir yerleştirme zamanı.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

TimeSetupMsc

Emir yerleştirme zamanını 01.01.1970 tarihinden itibaren milisaniyeler cinsinden alır.

```
ulong  TimeSetupMsc() const
```

Dönüş değeri

Emir yerleştirme zamanı (01.01.1970 tarihinden itibaren milisaniyeler cinsinden).

Not

Emir, önceden Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçiliş olmalıdır.

OrderType

Emir tipini alır.

```
ENUM_ORDER_TYPE  OrderType()
```

Dönüş değeri

Emir tipi (ENUM_ORDER_TYPE sayımının değerlerinden biri).

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

TypeDescription

Emir tipini dizgi biçiminde alır.

```
string  TypeDescription() const
```

Dönüş değeri

Dizgi biçiminde emir tipi.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

State

Emir durumunu alır.

```
ENUM_ORDER_STATE  State() const
```

Dönüş değeri

Emir durumu (ENUM_ORDER_STATE sayımının değerlerinden biri).

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

StateDescription

Emir durumunu dizgi biçiminde alır.

```
string  StateDescription() const
```

Dönüş değeri

Dizgi biçiminde emir durumu.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

TimeExpiration

Emir zaman-aşımı süresini alır.

```
datetime  TimeExpiration() const
```

Dönüş değeri

Yerleştirme anında belirtilen emir zaman-aşımı süresi.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

TimeDone

Emir uygulama veya iptal zamanını alır.

```
datetime  TimeDone() const
```

Dönüş değeri

Emir uygulama veya iptal zamanı.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

TimeDoneMsc

Emir uygulama veya iptal zamanını 01.01.1970 tarihinden itibaren milisaniyeler cinsinden alır.

```
ulong  TimeDoneMsc() const
```

Dönüş değeri

01.01.1970 tarihinden itibaren milisaniyeler cinsinden emir uygulama veya iptal zamanı.

Not

Emir, önceden Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçiliş olmalıdır.

TypeFilling

Emir karşılama türünü alır.

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

Dönüş değeri

Emir türü (ENUM_ORDER_TYPE_FILLING sayımının değerlerinden biri).

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

TypeFillingDescription

Emir türünü dizgi biçiminde alır.

```
string  TypeFillingDescription() const
```

Dönüş değeri

Dizgi biçiminde, emir karşılama türü.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

TypeTime

Zaman-aşımı anında emir tipini alır.

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

Dönüş değeri

Zaman-aşımı anındaki emir türü.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

TypeTimeDescription

Zaman-aşımı anında emir tipini dizgi şeklinde alır.

```
string  TypeTimeDescription() const
```

Dönüş değeri

Dizgi biçiminde, zaman-aşımı anındaki emir tipi.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

Magic

Emri yerleştiren Uzmanın tanımlayıcısını alır.

```
long  Magic() const
```

Dönüş değeri

Emri yerleştiren Uzmanın tanımlayıcısı.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

PositionId

Pozisyonun tanımlayıcısını alır.

```
long  PositionId() const
```

Dönüş değeri

Emre konu olan pozisyonun tanımlayıcısı.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

VolumeInitial

Emrin başlangıç hacim değerini alır.

```
double  VolumeInitial() const
```

Dönüş değeri

Emrin başlangıç hacim değeri.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

VolumeCurrent

Emrin karşılanmayan hacim miktarını alır.

```
double  VolumeCurrent() const
```

Dönüş değeri

Emrin karşılanmamış hacim miktarı (kalan).

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

PriceOpen

Emir fiyatını alır.

```
double  PriceOpen() const
```

Dönüş değeri

Emir yerleştirme fiyatını alır.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

StopLoss

Emrin Stop Loss fiyatını alır.

```
double  StopLoss() const
```

Dönüş değeri

Emrin Stop Loss fiyatı.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

TakeProfit

Emrin Take Profit fiyatını alır.

```
double  TakeProfit() const
```

Dönüş değeri

Emrin Take Profit fiyatı.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

PriceCurrent

Emir sembolünün mevcut fiyatını alır

```
double  PriceCurrent() const
```

Dönüş değeri

Emir sembolünün mevcut fiyatı.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

PriceStopLimit

Ayarlanan limit emrinin fiyatını alır.

```
double  PriceStopLimit() const
```

Dönüş değeri

Ayarlanan limit emrinin fiyatı.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

Symbol

Emir sembolünün ismini alır.

```
string  Symbol() const
```

Dönüş değeri

Emir sembolünün ismi.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

Comment

Emir yorumunu alır.

```
string  Comment() const
```

Dönüş değeri

Emir yorumu.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

InfoInteger

Belirtilen tamsayı tipli özelliğin değerini alır.

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,     // özellik tanımlayıcı
   long&                        var          // değişken referansı
   ) const
```

Parametreler

prop_id

[in] Tamsayı tipli özelliğin tanımlayıcısı (ENUM_ORDER_PROPERTY_INTEGER sayımının değerlerinden biri).

var

[out] Sonucu yerleştirmek için kullanılacak long tipli değişkenin referansı.

Dönüş değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

InfoDouble

Belirtilen double tipli özelliğin değerini alır.

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,      // özellik tanımlayıcı
   double&                     var           // değişken referansı
   ) const
```

Parametreler

prop_id

[in] double tipli özellik değeri (ENUM_ORDER_PROPERTY_DOUBLE sayımının değerlerinden biri).

var

[out] Sonucu yerleştirmek için kullanılacak olan double tipli değişkenin referansı.

Dönüş değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

InfoString

Belirtilen string tipli özelliğin değerini alır.

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,      // özellik tanımlayıcı
   string&                     var           // değişken referansı
   ) const
```

Parametreler

prop_id

[in] Dizgi biçimli özelliğin tanımlayıcısı.

var

[out] Sonucu yerleştirmek için kullanılacak string tipli değişkenin referansı.

Dönüş değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Not

Emir, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

StoreState

Emir parametrelerini kaydeder.

```
void  StoreState()
```

Dönüş değeri

Yok.

CheckState

Mevcut parametreleri kaydedilen parametrelerle karşılaştırır.

```
bool  CheckState()
```

Dönüş değeri

true - emir parametreleri StoreState() son çağrısından sonra değiştirilmişse, aksi durumda - false.

Select

Özelliklerine erişebilmek için fiş kullanarak bir emir seçer.

```
bool  Select(
   ulong  ticket   // emir fişi
   )
```

Dönüş değeri

Başarılı ise 'true', emir seçilemezse 'false'.

SelectByIndex

Özelliklerine erişebilmek için indis kullanarak bir emir seçer.

```
bool  SelectByIndex(
   int  index   // emir indisi
   )
```

Dönüş değeri

Başarılı ise 'true', emir seçilemezse 'false'.

CHistoryOrderInfo

CHistoryOrderInfo sınıfı, geçmiş emirlerin özelliklerine kolay erişim için tasarlanmıştır.

Açıklama

CHistoryOrderInfo sınıfı, geçmiş emirlerin özelliklerine erişim sağlar.

Bildirim

```
class CHistoryOrderInfo : public CObject
```

Başlık

```
#include <Trade\HistoryOrderInfo.mqh>
```

Kalıtım hiyerarşisi

CObject

CHistoryOrderInfo

Gruplarına göre sınıf yöntemleri

Tamsayı tipli özelliklere erişim için	
TimeSetup	Emir yerleştirme zamanını alır
TimeSetupMsc	Emir yerleştirme zamanını 01.01.1970 tarihinden itibaren milisaniyeler cinsinden alır
OrderType	Emir tipini alır
OrderTypeDescription	Emir tipini dizgi biçiminde alır
State	Emir durumunu alır
StateDescription	Emir durumunu dizgi biçiminde alır
TimeExpiration	Emir zaman-aşımı değerini alır
TimeDone	Emir uygulama veya iptal zamanını alır
TimeDoneMsc	Emir uygulama veya iptal zamanını 01.01.1970 tarihinden itibaren milisaniyeler cinsinden alır
TypeFilling	Emir uygulama türünü alır
TypeFillingDescription	Emir uygulama türünü dizgi biçiminde alır
TypeTime	Zaman-aşımı anında emir tipini alır
TypeTimeDescription	Zaman-aşımı anında emir tipini dizgi şeklinde alır
Magic	Emri yerleştiren Uzmanın tanımlayıcısını alır
PositionId	Pozisyonun tanımlayıcısını alır

Tamsayı tipli özelliklere erişim için	
double tipli özelliklere erişim	
VolumeInitial	Emrin başlangıç hacim değerini alır
VolumeCurrent	Emrin karşılanmamış hacim değerini alır
PriceOpen	Emir fiyatını alır
StopLoss	Emrin Stop Loss fiyatını alır
TakeProfit	Emrin Take Profit fiyatını alır
PriceCurrent	Emir sembolünün mevcut fiyatını alır
PriceStopLimit	Ayarlanan limit emrinin fiyatını alır
Metin özelliklerine erişim	
Symbol	Emir sembolünü alır
Comment	Emir yorumunu alır
MQL5 API fonksiyonlarına erişim	
InfoInteger	Belirtilen tamsayı tipli özelliğin değerini alır
InfoDouble	Belirtilen double tipli özelliğin değerini alır
InfoString	Belirtilen string tipli özelliğin değerini alır
Seçim	
Ticket	Emrin fişini alır/ Emri seçer
SelectByIndex	Emri indis ile seçer

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

TimeSetup

Emir yerleştirme zamanını alır.

```
datetime  TimeSetup() const
```

Dönüş değeri

Emir yerleştirme zamanı.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

TimeSetupMsc

Emir yerleştirme zamanını 01.01.1970 tarihinden itibaren milisaniyeler cinsinden alır.

```
ulong  TimeSetupMsc() const
```

Dönüş değeri

Emir yerleştirme zamanı (01.01.1970 tarihinden itibaren milisaniyeler cinsinden).

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

OrderType

Emir tipini alır.

```
ENUM_ORDER_TYPE  OrderType() const
```

Dönüş değeri

Emir tipi (ENUM_ORDER_TYPE sayımının değerlerinden biri).

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

TypeDescription

Emir tipini dizgi biçiminde alır.

```
string  TypeDescription() const
```

Dönüş değeri

Dizgi biçiminde emir tipi.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

State

Emir durumunu alır.

```
ENUM_ORDER_STATE  State() const
```

Dönüş değeri

Emir durumu (ENUM_ORDER_STATE sayımının değerlerinden biri).

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

StateDescription

Emir durumunu dizgi biçiminde alır.

```
string  StateDescription() const
```

Dönüş değeri

Dizgi biçiminde emir durumu.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

TimeExpiration

Emir zaman-aşımı değerini alır.

```
datetime  TimeExpiration() const
```

Dönüş değeri

Emir oluşturma sırasında tanımlanan zaman-aşımı değeri.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

TimeDone

Emir uygulama veya iptal zamanını alır.

```
datetime  TimeDone() const
```

Dönüş değeri

Emir uygulama veya iptal zamanı.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

TimeDoneMsc

Emir uygulama veya iptal zamanını 01.01.1970 tarihinden itibaren milisaniyeler cinsinden alır.

```
ulong  TimeDoneMsc() const
```

Dönüş değeri

Emir uygulama veya iptal zamanını (01.01.1970 tarihinden itibaren milisaniyeler cinsinden).

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

TypeFilling

Emir uygulama türünü alır.

```
ENUM_ORDER_TYPE_FILLING  TypeFilling() const
```

Dönüş değeri

Emir uygulama türü (ENUM_ORDER_TYPE_FILLING sayımının değerlerinden biri).

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

TypeFillingDescription

Emir uygulama türünü dizgi biçiminde alır.

```
string  TypeFillingDescription() const
```

Dönüş değeri

Dizgi biçiminde, emir uygulama türü.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

TypeTime

Zaman-aşımı anında emir tipini alır.

```
ENUM_ORDER_TYPE_TIME  TypeTime() const
```

Dönüş değeri

Zaman-aşımı anındaki emir tipi (ENUM_ORDER_TYPE_TIME sayımının değerlerinden biri).

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

TypeTimeDescription

Zaman-aşımı anında emir tipini dizgi şeklinde alır.

```
string  TypeTimeDescription() const
```

Dönüş değeri

Dizgi biçiminde, zaman-aşımı anındaki emir tipi.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

Magic

Emri yerleştiren Uzman Danışmanın tanımlayıcısını alır

```
long  Magic() const
```

Dönüş değeri

Emri yerleştiren Uzman Danışmanın tanımlayıcısı.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

PositionId

Pozisyonun tanımlayıcısını alır.

```
long  PositionId() const
```

Dönüş değeri

Emre konu olan pozisyonun tanımlayıcısı.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

VolumeInitial

Emrin başlangıç hacim değerini alır.

```
double  VolumeInitial() const
```

Dönüş değeri

Emrin başlangıç hacim değeri.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

VolumeCurrent

Emrin karşılanmayan hacim miktarını alır.

```
double  VolumeCurrent() const
```

Dönüş değeri

Emrin karşılanmamış hacim miktarı (kalan).

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

PriceOpen

Emir fiyatını alır.

```
double  PriceOpen() const
```

Dönüş değeri

Emir yerleştirme fiyatını alır.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

StopLoss

Emrin Stop Loss fiyatını alır.

```
double  StopLoss() const
```

Dönüş değeri

Emrin Stop Loss fiyatı.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

TakeProfit

Emrin Take Profit fiyatını alır.

```
double  TakeProfit() const
```

Dönüş değeri

Emrin Take Profit fiyatı.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

PriceCurrent

Emir sembolünün mevcut fiyatını alır.

```
double  PriceCurrent() const
```

Dönüş değeri

Emir sembolünün mevcut fiyatı.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

PriceStopLimit

Emrin stop limit fiyatını alır.

```
double  PriceStopLimit() const
```

Dönüş değeri

Emrin stop limit fiyatı.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

Symbol

Emir sembolünün ismini alır.

```
string  Symbol() const
```

Dönüş değeri

Emir sembolünün ismi.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

Comment

Emir yorumunu alır.

```
string  Comment() const
```

Dönüş değeri

Emir yorumu.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

InfoInteger

Belirtilen tamsayı tipli özelliğin değerini alır.

```
bool  InfoInteger(
   ENUM_ORDER_PROPERTY_INTEGER  prop_id,     // özellik tanımlayıcı
   long&                        var          // değişken referansı
   ) const
```

Parametreler

prop_id

[in] Tamsayı tipli özelliğin tanımlayıcısı (ENUM_ORDER_PROPERTY_INTEGER sayımının değerlerinden biri).

var

[out] Sonucu yerleştirmek için kullanılacak long tipli değişkenin referansı.

Dönüş değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

InfoDouble

Belirtilen double tipli özelliğin değerini alır.

```
bool  InfoDouble(
   ENUM_ORDER_PROPERTY_DOUBLE  prop_id,      // özellik tanımlayıcı
   double&                     var           // değişken referansı
   ) const
```

Parametreler

prop_id

[in] double tipli özellik değeri (ENUM_ORDER_PROPERTY_DOUBLE sayımının değerlerinden biri).

var

[out] Sonucu yerleştirmek için kullanılacak olan double tipli değişkenin referansı.

Dönüş değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

InfoString

Belirtilen string tipli özelliğin değerini alır.

```
bool  InfoString(
   ENUM_ORDER_PROPERTY_STRING  prop_id,      // özellik tanımlayıcı
   string&                     var           // değişken referansı
   ) const
```

Parametreler

prop_id

[in] Dizgi biçimli özelliğin tanımlayıcısı (ENUM_ORDER_PROPERTY_STRING sayımının değerlerinden biri).

var

[out] Sonucu yerleştirmek için kullanılacak string tipli değişkenin referansı.

Dönüş değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Not

Geçmiş emir, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemleri ile seçilmelidir.

Ticket (Get Yöntemi)

Emir fişini alır.

```
ulong  Ticket() const
```

Dönüş değeri

Emir fişi.

Ticket (Set Yöntemi)

Daha sonraki işlemler için emri seçer.

```
void  Ticket(
   ulong  ticket      // emir fişi
   )
```

Parametreler

ticket

[in] Emir fişi.

SelectByIndex

Özelliklerine erişebilmek için indis kullanarak bir emir seçer.

```
bool  SelectByIndex(
   int  index   // emir indisi
   )
```

Dönüş değeri

Başarılı ise 'true', emir seçilemezse 'false'.

CPositionInfo

CPositionInfo, açık durumdaki pozisyonların özelliklerine kolay erişim sağlamak için geliştirilmiş bir sınıftır.

Açıklama

CPositionInfo sınıfı, açık durumdaki pozisyonların özelliklerine kolay erişim sağlar.

Bildirim

```
class CPositionInfo : public CObject
```

Başlık

```
#include <Trade\PositionInfo.mqh>
```

Kalıtım hiyerarşisi

CObject

CPositionInfo

Gruplarına göre sınıf yöntemleri

Tamsayı tipli özelliklere erişim için	
Time	Pozisyon açılış zamanını alır
TimeMsc	Pozisyon açılış zamanını 01.01.1970 tarihinden itibaren milisaniyeler cinsinden alır
TimeUpdate	Pozisyon değişimi zamanını 01.01.1970 tarihinden itibaren saniyeler cinsinden alır
TimeUpdateMsc	Pozisyon değişimi zamanını 01.01.1970 tarihinden itibaren milisaniyeler cinsinden alır
PositionType	Pozisyon tipini alır.
TypeDescription	Pozisyon tipini dizgi biçiminde alır
Magic	Pozisyonu açan Uzmanın tanımlayıcısını alır
Tanıtıcı	Pozisyonun tanımlayıcısını alır
double tipli özelliklere erişim	
Volume	Pozisyonun hacim değerini alır
PriceOpen	Pozisyonun açılış fiyatını alır
StopLoss	Pozisyonun Zarar Durdur fiyatını alır
TakeProfit	Pozisyonun Kar Al fiyatını alır

Tamsayı tipli özelliklere erişim için	
PriceCurrent	Pozisyon sembolünün mevcut fiyatını alır
Commission	Pozisyona konu olan komisyon miktarını alır
Swap	Pozisyona konu olan swap değerini alır
Profit	Pozisyonun mevcut kar miktarını alır
Metin özelliklerine erişim	
Symbol	Pozisyonda kullanılan sembolün ismini alır
Comment	Pozisyon yorumunu alır
MQL5 API fonksiyonlarına erişim	
InfoInteger	Belirtilen tamsayı tipli özelliğin değerini alır
InfoDouble	Belirtilen double tipli özelliğin değerini alır
InfoString	Belirtilen dizgi tipli özelliğin değerini alır.
Seçim	
Select	Pozisyonu seçer
SelectByIndex	Pozisyonu indis kullanarak seçer
SelectByMagic	Belirtilen sembol ismine ve tanıtıcı değere (magic number) göre bir pozisyon seçer
SelectByTicket	Pozisyonu fiş değerine göre seçer
Durum	
StoreState	Pozisyon parametrelerini kaydeder
CheckState	Mevcut parametreleri kaydedilen parametrelerle karşılaştırır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Time

Pozisyon açılış zamanını alır.

```
datetime  Time() const
```

Dönüş değeri

Pozisyon açılış zamanı.

Not

Pozisyon, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

TimeMsc

Pozisyon açılış zamanını 01.01.1970 tarihinden itibaren milisaniyeler cinsinden alır.

```
ulong  TimeMsc() const
```

Dönüş değeri

01.01.1970 tarihinden itibaren milisaniyeler cinsinden Pozisyon açılış zamanı.

Not

Pozisyon, önceden Select (sembol kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçiliş olmalıdır.

TimeUpdate

Pozisyon değişimi zamanını 01.01.1970 tarihinden itibaren saniyeler cinsinden alır.

```
datetime  TimeUpdate() const
```

Dönüş değeri

Pozisyon değişimi zamanı (01.01.1970 tarihinden itibaren saniyeler cinsinden).

Not

Pozisyon, önceden Select (sembol kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçiliş olmalıdır.

TimeUpdateMsc

Pozisyon değişimi zamanını 01.01.1970 tarihinden itibaren milisaniyeler cinsinden alır.

```
ulong  TimeUpdateMsc() const
```

Dönüş değeri

Pozisyon değişimi zamanı (01.01.1970 tarihinden itibaren milisaniyeler cinsinden).

Not

Pozisyon, önceden Select (sembol kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçiliş olmalıdır.

PositionType

Pozisyon tipini alır.

```
ENUM_POSITION_TYPE  PositionType() const
```

Dönüş değeri

Pozisyon tipi (ENUM_POSITION_TYPE sayımının değerlerinden biri).

Not

Pozisyon, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

TypeDescription

Pozisyon tipini dizgi biçiminde alır.

```
string  TypeDescription() const
```

Dönüş değeri

Dizgi biçimli pozisyon tipi.

Not

Pozisyon, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

Magic

Pozisyonu açan Uzmanın tanımlayıcısını alır.

```
long  Magic() const
```

Dönüş değeri

Pozisyonu açan Uzmanın tanımlayıcısı.

Not

Pozisyon, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

Identifier

Pozisyonun tanımlayıcısını alır.

```
long  Identifier() const
```

Dönüş değeri

Pozisyonun tanımlayıcısı.

Not

Pozisyon, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

Volume

Pozisyonun hacim değerini alır.

```
double  Volume() const
```

Dönüş değeri

Pozisyonun hacim değeri.

Not

Pozisyon, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

PriceOpen

Pozisyonun açılış fiyatını alır.

```
double  PriceOpen() const
```

Dönüş değeri

Pozisyonun açılış fiyatı.

Not

Pozisyon, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

StopLoss

Pozisyonun Stop Loss fiyatını alır.

```
double  StopLoss() const
```

Dönüş değeri

Pozisyonun Stop Loss fiyatı.

Not

Pozisyon, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

TakeProfit

Pozisyonun Take Profit fiyatını alır.

```
double  TakeProfit() const
```

Dönüş değeri

Pozisyonun Take Profit fiyatı.

Not

Pozisyon, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

PriceCurrent

Pozisyon sembolünün mevcut fiyatını alır.

```
double  PriceCurrent() const
```

Dönüş değeri

Pozisyon sembolünün mevcut fiyatı.

Commission

Pozisyona konu olan komisyon miktarını alır.

```
double  Commission() const
```

Dönüş değeri

Pozisyona konu olan komisyon miktarını - mevduat dövizinin cinsinden - alır.

Not

Pozisyon, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

Swap

Pozisyona konu olan swap değerini alır.

```
double  Swap() const
```

Dönüş değeri

Pozisyona konu olan swap değeri (mevduat dövizinin cinsinden).

Not

Pozisyon, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

Profit

Pozisyonun mevcut kar miktarını alır.

```
double  Profit() const
```

Dönüş değeri

Pozisyonun mevcut kar miktarı (mevduat dövizinin cinsinden).

Not

Pozisyon, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

Symbol

Pozisyonda kullanılan sembolün ismini alır.

```
string  Symbol() const
```

Dönüş değeri

Pozisyonda kullanılan sembolün ismi.

Not

Pozisyon, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

Comment

Pozisyon yorumunu alır.

```
string  Comment() const
```

Dönüş değeri

Pozisyon yorumunu.

Not

Pozisyon, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

InfoInteger

Belirtilen tamsayı tipli özelliğin değerini alır.

```
bool  InfoInteger(
   ENUM_POSITION_PROPERTY_INTEGER  prop_id,      // özellik tanımlayıcı
   long&                           var           // değişken referansı
   ) const
```

Parametreler

prop_id

[in] Tamsayı tipli özelliğin tanımlayıcısı (ENUM_POSITION_PROPERTY_INTEGER sayımının değerlerinden biri).

var

[out] Sonucu yerleştirmek için kullanılacak long tipli değişkenin referansı.

Dönüş değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Not

Pozisyon, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

InfoDouble

Belirtilen double tipli özelliğin değerini alır.

```
bool  InfoDouble(
   ENUM_POSITION_PROPERTY_DOUBLE  prop_id,      // özellik tanımlayıcı
   double&                        var           // değişken referansı
   ) const
```

Parametreler

prop_id

[in] double tipli özelliğin tanımlayıcısı (ENUM_POSITION_PROPERTY_DOUBLE sayımının değerlerinden biri).

var

[in] Sonucu yerleştirmek için kullanılacak olan double tipli değişkenin referansı.

Dönüş değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Not

Pozisyon, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

InfoString

Belirtilen string tipli özelliğin değerini alır.

```
bool  InfoString(
   ENUM_POSITION_PROPERTY_STRING  prop_id,      // özellik tanımlayıcı
   string&                        var           // değişken referansı
   ) const
```

Parametreler

prop_id

[in] Dizgi biçimli özelliğin tanımlayıcısı (ENUM_POSITION_PROPERTY_STRING sayımının değerlerinden biri).

var

[out] Sonucu yerleştirmek için kullanılacak string tipli değişkenin referansı.

Dönüş değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Not

Pozisyon, Select (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemi ile seçilmelidir.

Select

Daha ileri işlemler için pozisyonu seçer.

```
bool  Select(
   const string  symbol      // sembol
   )
```

Parametreler

symbol

[in] Pozisyon seçimi için kullanılacak sembol.

SelectByIndex

Özelliklerine erişebilmek için indis kullanarak bir pozisyon seçer.

```
bool  SelectByIndex(
   int  index   // pozisyon indisi
   )
```

Dönüş değeri

Başarılı ise 'true', pozisyon seçilemezse 'false'.

SelectByMagic

Finansal enstrüman ismine ve tanıtıcı değere (magic number) göre bir pozisyon seçer.

```
bool  SelectByMagic(
   const string  symbol, // sembol ismi
   const ulong   magic   // tanıtıcı değer
   );
```

Parametreler

symbol

[in] Sembol ismi.

magic

[in] Pozisyonun tanıtıcı değeri (Sihirli sayı).

Dönüş Değeri

Başarılı ise 'true', pozisyon seçilemezse 'false'.

SelectByTicket

Pozisyonu fiş değerine göre seçer.

```
bool  SelectByTicket(
   ulong  ticket      // pozisyon fişi
   );
```

Parametreler

ticket

[in] Pozisyon fişi.

Dönüş değeri

İşlem başarılı ise 'true', aksi durumda 'false'.

StoreState

Pozisyon parametrelerini kaydeder.

```
void  StoreState()
```

Dönüş değeri

Yok.

CheckState

Mevcut parametreleri kaydedilen parametrelerle karşılaştırır.

```
bool  CheckState()
```

Dönüş değeri

true - pozisyon parametreleri StoreState() son çağrısından sonra değiştirilmişse, aksi durumda - false.

CDealInfo

CDealInfo sınıfı, işlem özelliklerine kolay erişim için tasarlanmıştır.

Açıklama

CDealInfo sınıfı, işlem özelliklerine kolay erişim sağlar.

Bildirim

```
class CDealInfo : public CObject
```

Başlık

```
#include <Trade\DealInfo.mqh>
```

Kalıtım hiyerarşisi

CObject

CDealInfo

Gruplarına göre sınıf yöntemleri

Tamsayı tipli özelliklere erişim için	
Order	İşlemin ugulandığı emri alır.
Time	İşlemin uygulama zamanını alır
TimeMsc	işlemin uygulama zamanını 01.01.1970 tarihinden itibaren milisaniyeler cinsinden alır
DealType	İşlem tipini alır
TypeDescription	İşlem tipini dizgi biçiminde alır
Entry	işlem yönünü alır
EntryDescription	İşlem yönünü dizgi cinsinden alır
Magic	İşlemi gerçekleştiren Uzmanın tanımlayıcısını alır
PositionId	İşleme konu olan pozisyonun tanımlayıcısını alır.
double tipli özelliklere erişim	
Volume	İşlemin hacim değerini alır
Price	İşlemin fiyatını alır
Commision	İşleme konu olan komisyon miktarını alır
Swap	Pozisyon kapatıldığında swap değerini alır
Profit	İşlemin finansal sonucunu alır

Tamsayı tipli özelliklere erişim için	
Metin özelliklerine erişim	
Symbol	İşlem sembolünü alır
Comment	İşlem yorumunu alır
MQL5 API fonksiyonlarına erişim	
InfoInteger	Belirtilen tamsayı tipli özelliğin değerini alır
InfoDouble	Belirtilen double tipli özelliğin değerini alır
InfoString	Belirtilen string tipli özelliğin değerini alır
Seçim	
Ticket	İşlemin fişini alır/ işlemi seçer
SelectByIndex	İşlemi indis kullanarak seçer

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Order

İşlemin ugulandığı emri alır.

```
long  Order() const
```

Dönüş değeri

İşlemin ugulandığı emir.

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

Time

İşlemin uygulama zamanını alır.

```
datetime  Time() const
```

Dönüş değeri

İşlemin uygulama zamanı.

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

TimeMsc

işlemin uygulama zamanını 01.01.1970 tarihinden itibaren milisaniyeler cinsinden alır.

```
ulong  TimeMsc() const
```

Dönüş değeri

01.01.1970 tarihinden itibaren milisaniyeler cinsinden, işlemin uygulama zamanı.

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmiş olmalıdır.

DealType

İşlem tipini alır.

```
ENUM_DEAL_TYPE  DealType() const
```

Dönüş değeri

İşlem tipi (ENUM_DEAL_TYPE sayımının değerlerinden biri).

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

TypeDescription

İşlem tipini dizgi biçiminde alır.

```
string  TypeDescription() const
```

Dönüş değeri

Dizgi cinsinden işlem tipi.

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

Entry

işlem yönünü alır.

```
ENUM_DEAL_ENTRY  Entry() const
```

Dönüş değeri

İşlem yönü (ENUM_DEAL_ENTRY sayımının değerlerinden biri).

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

EntryDescription

İşlem yönünü dizgi cinsinden alır.

```
string  EntryDescription() const
```

Dönüş değeri

Dizgi cinsinden, işlem yönü.

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

Magic

İşlemi gerçekleştiren Uzmanın tanımlayıcısını alır.

```
long  Magic() const
```

Dönüş değeri

İşlemi gerçekleştiren Uzmanın tanımlayıcısı.

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

PositionId

İşleme konu olan pozisyonun tanımlayıcısını alır.

```
long  PositionId() const
```

Dönüş değeri

İşleme konu olan pozisyonun tanımlayıcısı.

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

Volume

İşlemin hacim değerini alır.

```
double  Volume() const
```

Dönüş değeri

İşlemin hacim değeri.

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

Price

İşlemin fiyatını alır.

```
double  Price() const
```

Dönüş değeri

İşlemin fiyatı.

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

Commission

İşleme konu olan komisyon miktarını alır.

```
double  Commission() const
```

Dönüş değeri

İşleme konu olan komisyon miktarı.

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

Swap

Pozisyon kapatıldığında swap değerini alır

```
double  Swap() const
```

Dönüş değeri

Pozisyon kapatıldığı zamanki swap değeri.

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

Profit

İşlemin finansal sonucunu alır.

```
double  Profit() const
```

Dönüş değeri

İşlemin finansal sonucu (mevduat dövizi cinsinden).

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

Symbol

İşlem sembolünün ismini alır.

```
string  Symbol() const
```

Dönüş değeri

İşlem sembolünün ismi.

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

Comment

İşlem yorumunu alır.

```
string  Comment() const
```

Dönüş değeri

İşlem yorumu.

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

InfoInteger

Belirtilen tamsayı tipli özelliğin değerini alır.

```
bool  InfoInteger(
   ENUM_DEAL_PROPERTY_INTEGER  prop_id,      // özellik tanımlayıcı
   long&                       var           // değişken referansı
   ) const
```

Parametreler

prop_id

[in] Tamsayı tipli özelliğin tanımlayıcısı (ENUM_DEAL_PROPERTY_INTEGER sayımının değerlerinden biri).

var

[out] Sonucu yerleştirmek için kullanılacak long tipli değişkenin referansı.

Dönüş değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

InfoDouble

Belirtilen double tipli özelliğin değerini alır.

```
bool  InfoDouble(
   ENUM_DEAL_PROPERTY_DOUBLE  prop_id,      // özellik tanımlayıcı
   double&                    var           // referans değişken
   ) const
```

Parametreler

prop_id

[in] double tipli özelliğin tanımlayıcısı (ENUM_DEAL_PROPERTY_DOUBLE sayımının değerlerinden biri).

var

[in] Sonucu yerleştirmek için kullanılacak olan double tipli değişkenin referansı.

Dönüş değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

InfoString

Belirtilen string tipli özelliğin değerini alır.

```
bool  InfoString(
   ENUM_DEAL_PROPERTY_STRING  prop_id,      // özellik tanımlayıcı
   string&                    var           // değişken referansı
   ) const
```

Parametreler

prop_id

[in] Dizgi biçimli özelliğin tanımlayıcısı (ENUM_DEAL_PROPERTY_STRING sayımının değerlerinden biri).

var

[out] Sonucu yerleştirmek için kullanılacak string tipli değişkenin referansı.

Dönüş değeri

Başarılı ise 'true', özellik değeri alınamazsa 'false'.

Not

İşlem, Ticket (fiş kullanarak) veya SelectByIndex (indis kullanarak) yöntemiyle seçilmelidir.

Ticket (Get Yöntemi)

İşlemin fişini alır.

```
ulong  Ticket() const
```

Dönüş değeri

İşlem fişi.

Ticket (Set Yöntemi)

Daha ileri işlemler için pozisyonu seçer.

```
void  Ticket(
   ulong  ticket      // fiş
   )
```

Parametreler

ticket

[in] İşlem fişi.

SelectByIndex

Özelliklerine erişebilmek için indis kullanarak bir işlem seçer.

```
bool  SelectByIndex(
   int  index   // emir indisi
   )
```

Dönüş değeri

Başarılı ise 'true', işlem seçilemezse 'false'.

CTrade

CTrade sınıfı alım-satım fonksiyonlarına kolay erişim için tasarlanmıştır.

Açıklama

CTrade sınıfı alım-satım fonksiyonlarına kolay erişim sağlar.

Bildirim

```
class CTrade : public CObject
```

Başlık

```
#include <Trade\Trade.mqh>
```

Kalıtım hiyerarşisi

CObject

CTrade

İlk nesil

CExpertTrade

Gruplarına göre sınıf yöntemleri

Parametrelerin ayarlanması	
LogLevel	Günlük oluşturma seviyesini ayarlar
SetExpertMagicNumber	Uzmanın tanımlayıcısını ayarlar
SetDeviationInPoints	İzin verilen sapma değerini aayarlar
SetTypeFilling	Emir karşılama türünü ayarlar
SetTypeFillingBySymbol	Belirtilen sembol ayarlarına göre emir uygulama yöntemini ayarlar
SetAsyncMode	Alım-satım işlemi için asenkron modu ayarlar
SetMarginMode	Mevcut hesap ayarlarına göre teminat hesaplama modunu değiştirir
Emir işlemleri	
OrderOpen	Belirtilen parametrelerle bir bekleyen emir yerleştirir
OrderModify	Bekleyen emir parametrelerini değiştirir
OrderDelete	Bekleyen emri siler
Pozisyon işlemleri	
PositionOpen	Belirtilen parametrelerle bir pozisyon açar

Parametrelerin ayarlanması	
PositionModify	Belirtilen sembol ismine veya fişine göre pozisyonun parametrelerini değiştirir
PositionClose	Belirtilen sembol için bir açık pozisyonu kapatır.
PositionClosePartial	Belirtilen sembol üzerindeki veya belirtilen fişe sahip olan pozisyonu kısmen kapatır
PositionCloseBy	Pozsyonu belirtilen ters pozisyonun fişine göre kapatır
Ek yöntemler	
Alış	Belirtilen parametrelerle bir uzun pozisyon açar
Satış	Belirtilen parametrelerle bir kısa pozisyon açar
BuyLimit	Belirtilen parametrelerle Limit Alış tipinde bir bekleyen emir yerleştirir
BuyStop	Belirtilen parametrelerle Stop Alış tipinde bir bekleyen emir yerleştirir
SellLimit	Belirtilen parametrelerle Limit Satış tipinde bir bekleyen emir yerleştirir
SellStop	Belirtilen parametrelerle Stop Satış tipinde bir bekleyen emir yerleştirir
Son isteğin parametrelerine erişim	
Request	Son istek yapısının kopyasını alır
RequestAction	Alım-satım işlem tipini alır
RequestActionDescription	Alım-satım işlem tipini dizgi biçiminde alır
RequestMagic	Uzman Danışman tanımlayıcısını (magic number) alır
RequestOrder	Son istekte kullanılmış olan emir fişini alır
RequestSymbol	Son istekte kullanılmış olan sembolün ismini alır
RequestVolume	Son istekte kullanılmış olan işlem hacmini lot cinsinden alır
RequestPrice	Son istekte kullanılmış olan fiyatı alır
RequestStopLimit	Son istekte kullanılmış olan Stop Limit tipli bekleyen emrin fiyatını alır
RequestSL	Son istekte kullanılmış olan Zarar Durdur fiyatını alır.
RequestTP	Son istekte kullanılmış olan Kar Al fiyatını alır
RequestDeviation	Son istekte kullanılmış olan fiyat sapması değerini alır
RequestType	Son istekte kullanılan emir tipini alır

Parametrelerin ayarlanması	
RequestTypeDescription	Son istekte kullanılmış olan emir tipini dizgi biçiminde alır
RequestTypeFilling	Son istekte kullanılmış olan emir karşılama türünü alır
RequestTypeFillingDescription	Son istekte kullanılmış olan emir karşılama türünü dizgi biçiminde alır
RequestTypeTime	Son istekte kullanılmış olan emir geçerlilik süresini alır
RequestTypeTimeDescription	Son istekte kullanılmış olan emir geçerlilik süresini dizgi biçiminde alır
RequestExpiration	Son istekte kullanılmış olan zaman-aşımı süresinin değerini alır
RequestComment	Son istekte kullanılmış olan emir yorumunu alır
RequestPosition	Pozisyonun fişini alır
RequestPositionBy	Ters pozisyonun fişini alır
Son isteğin denetim sonuçlarına erişim	
CheckResult	Son isteğin denetim sonucu yapısının kopyasını alır.
CheckResultRetcode	İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'retcode' alanının değerini alır
CheckResultRetcodeDescription	İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'retcode' (dönüş kodu) alanının dizgi biçimli açılamasını alır
CheckResultBalance	İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'balance' (bakiye) alanının değerini alır
CheckResultEquity	İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'equity' (varlık) alanının değerini alır
CheckResultProfit	Alım-satım işleminden sonra gerçekleşecek değişken kar değerini alır
CheckResultMargin	İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'margin' (teminat) alanının değerini alır
CheckResultMarginFree	İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'margin_free' (serbest teminat) alanının değerini alır
CheckResultMarginLevel	İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'margin_level' (teminat seviyesi) alanının değerini alır

Parametrelerin ayarlanması	
CheckResultComment	İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'comment' (yorum) alanının değerini alır
İstek uygulama sonuçlarına erişim	
Result	Son istek sonucu yapısının kopyasını alır
ResultRetcode	İstek sonucunun kodunu alır
ResultRetcodeDescription	İstek sonucunun kodunu dizgi biçiminde alır
ResultDeal	İşlem fişini alır
ResultOrder	Emir fişini alır
ResultVolume	Emir veya işlem hacmini alır
ResultPrice	Aracı kurum tarafından doğrulanan fiyatı alır
ResultBid	Geçerli teklif (bid) fiyatını alır
ResultAsk	Geçerli istek (ask) fiyatını alır
ResultComment	Aracı kurumun yorumunu alır
Yardımcı yöntemler	
PrintRequest	Son istek parametrelerini günlüğe yazdırır
PrintResult	Son istek sonuçlarını günlüğe yazdırır
FormatRequest	Son istek parametreleriye biçimlendirilmiş bir dizgi hazırlar
FormatRequestResult	Son isteğin uygulama sonuçlarıyla biçimlendirilmiş bir dizgi hazırlar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

LogLevel

Mesajlar için günlük oluşturma seviyesini ayarlar.

```
void  LogLevel(
   ENUM_LOG_LEVELS  log_level      // günlükleme seviyesi
   )
```

Parametreler

log_level

[in] Günlük oluşturma seviyesi.

Dönüş değeri

Yok.

Not

LOG_LEVEL_NO ve daha azı, mesaj görüntülemeyi devre-dışı bırakır (optimizasyon modunda otomatik olarak ayarlanır). LOG_LEVEL_ERRORS hata mesajı görüntülemeyi etkinleştirir (varsayılan değer). LOG_LEVEL_ALL ve daha fazlası tüm mesajların görüntülenmesini etkinleştirir (sınama modunda otomatik olarak ayarlanır).

ENUM_LOG_LEVELS

Tanımlayıcı	Açıklama	Değer
LOG_LEVEL_NO	Mesaj görüntüleme devre-dışı	0
LOG_LEVEL_ERRORS	Sadece hata mesajları görüntülenir	1
LOG_LEVEL_ALL	Tüm mesajlar görüntülenir	2

SetExpertMagicNumber

Uzmanın tanımlayıcısını ayarlar.

```
void  SetExpertMagicNumber(
   ulong  magic      // tanımlayıcı
   )
```

Parametreler

magic

[in] Uzmanın yeni tanımlayıcısı.

Dönüş değeri

Yok.

SetDeviationInPoints

İzin verilen sapma değerini aayarlar.

```
void  SetDeviationInPoints(
   ulong  deviation      // sapma
   )
```

Parametreler

deviation

[in] İzin verilen sapma değeri.

Dönüş değeri

Yok.

SetTypeFilling

Emir karşılama türünü ayarlar.

```
void  SetTypeFilling(
   ENUM_ORDER_TYPE_FILLING  filling      // emir karşılama türü
   )
```

Parametreler

filling

[in] Emir karşılama türü (ENUM_ORDER_TYPE_FILLING sayımının değerlerinden biri).

Dönüş değeri

Yok.

SetTypeFillingBySymbol

Belirtilen sembol ayarlarına göre emir uygulama yöntemini ayarlar.

```
bool  SetTypeFillingBySymbol(
   const string   symbol      // sembol ismi
   )
```

Parametreler

symbol

[in] SYMBOL_FILLING_MODE hanesinde izin verilen uygulama yöntemi bulunan sembolün ismi.

Dönüş Değeri

Başarılı ise 'true', dolgu yöntemi tanımlama başarısız ise 'false'

Not

Sembol üzerinde SYMBOL_FILLING_FOK ve SYMBOL_FILLING_IOC yöntemlerine birlikte izin veriliyorsa, emir için ORDER_FILLING_FOK değeri ayarlanır.

SetAsyncMode

Alım-satım işlemi için asenkron modu ayarlar.

```
void  SetAsyncMode(
   bool  mode      // asenkron mod bayrağı
   )
```

Parametreler

mode

[in] Asenkron mod bayrağı.

Dönüş değeri

Yok.

Not

Bu mod, asenkron alım-satım işlemleri için kullanılır (gönderilen isteğe işlem sunucusunun cevabı beklenmeden yapılan işlem) (bakınız: OrderSendAsync).

SetMarginMode

Mevcut hesap ayarlarına göre teminat hesaplama modunu değiştirir.

```
void  SetMarginMode()
```

Dönüş Değeri

Yok.

Not

Teminat hesaplama modu ENUM_ACCOUNT_MARGIN_MODE ile belirtilir.

OrderOpen

Belirtilen parametrelerle bir bekleyen emir yerleştirir.

```
bool  OrderOpen(
   const string          symbol,          // sembol
   ENUM_ORDER_TYPE       order_type,      // emir tipi
   double                volume,          // emir hacmi
   double                limit_price,     // StopLimit fiyatı
   double                price,           // uygulama fiyatı
   double                sl,              // Stop Loss fiyatı
   double                tp,              // Take Profit fiyatı
   ENUM_ORDER_TYPE_TIME  type_time,       // zaman-aşımına göre yeni tip
   datetime              expiration,      // zaman aşımı
   const string          comment=""       // yorum
   )
```

Parametreler

symbol

[in] Alım-satım enstrümanının ismi.

order_type

[in] alım-satım işleminin tipi (ENUM_ORDER_TYPE sayımının değerlerinden biri).

volume

[in] İstenen emir hacmi.

limit_price

[in] StopLimit emrinin yerleştirileceği fiyat.

price

[in] Emrin uygulanması gereken fiyat.

sl

[in] Stop Loss için tetiklenme fiyatı.

tp

[in] Take Profit tetiklenme fiyatı.

type_time

[in] Zaman-aşımına göre yeni emir tipi (ENUM_ORDER_TYPE_TIME sayımının değerlerinden biri).

expiration

[in] Bekleyen emrin zaman-aşımı tarihi.

comment=""

[in] Emir yorumu.

Dönüş değeri

Temel yapılar başarıyla denetlenirse 'true', aksi durumda 'false'.

Not

OrderSend(...) yönteminin başarıyla sonuçlanması her zaman başarılı bir alım-satım işlemi uygulandığı anlamına gelmez. Bu yüzden, alım-satım isteğinin sonucunun (işlem sunucusunun dönüş kodu)ResultRetcode() yöntemi ve ResultOrder() yönteminin dönüş değeri kullanılarak denetlenmesi önemlidir.

OrderModify

Bekleyen emir parametrelerini değiştirir.

```
bool  OrderModify(
   ulong                 ticket,          // emir fişi
   double                price,           // uygulama fiyatı
   double                sl,              // Stop Loss fiyatı
   double                tp,              // Take Profit fiyatı
   ENUM_ORDER_TYPE_TIME  type_time,       // zaman-aşımına göre yeni tip
   datetime              expiration,      // zaman-aşımı
   double                stoplimit        // Limit emri fiyatı
   )
```

Parametreler

ticket

[in] Emir fişi.

price

[in] Emrin uygulanacağı yeni fiyat (veya değişim gereksizse eski değer).

sl

[in] Yeni Stop Loss değeri (veya değişim gereksizse eski değer).

tp

[in] Yeni Take Profit değeri (veya değişim gereksizse eski değer).

type_time

[in] Zaman-aşımına göre yeni emir tipi (veya değişim gereksizse eski değer), ENUM_ORDER_TYPE_TIME sayımının değerlerinden biri.

expiration

[in] Bekleyen emir için yeni zaman-aşımı süresi (veya değişim gereksizse eski değer).

stoplimit

[in] Piyasa fiyatı *price* değerine ulaştığında yeni bir Limit emri ayarlamak için kullanılacak yeni fiyat. Sadece StopLimit emirleri için belirtilir.

Dönüş değeri

Temel yapılar başarıyla denetlenirse 'true', aksi durumda 'false'.

Not

OrderModify(...) yönteminin başarıyla sonuçlanması her zaman başarılı bir alım-satım işlemi uygulandığı anlamına gelmez. Bu yüzden, alım-satım isteğinin sonucunun (işlem sunucusunun dönüş kodu) ResultRetcode() yöntemi kullanılarak denetlenmesi önemlidir.

OrderDelete

Bekleyen emri siler.

```
bool  OrderDelete(
   ulong  ticket      // emir fişi
   )
```

Parametreler

ticket

[in] Emir fişi.

Dönüş değeri

Temel yapılar başarıyla denetlenirse 'true', aksi durumda 'false'.

Not

OrderDelete(...) yönteminin başarıyla sonuçlanması her zaman başarılı bir alım-satım işlemi uygulandığı anlamına gelmez. Bu yüzden, alım-satım isteğinin sonucunun (işlem sunucusunun dönüş kodu) ResultRetcode() yöntemi kullanılarak denetlenmesi önemlidir.

PositionOpen

Belirtilen parametrelerle bir pozisyon açar.

```
bool  PositionOpen(
   const string     symbol,          // sembol
   ENUM_ORDER_TYPE  order_type,      // pozisyon açmak için emir tipi
   double           volume,          // pozisyon hacmi
   double           price,           // uygulama fiyatı
   double           sl,              // Stop Loss fiyatı
   double           tp,              // Take Profit fiyatı
   const string     comment=""       // yorum
   )
```

Parametreler

symbol

[in] Pozisyon açmak için kullanılacak enstrümanın ismi.

order_type

[in] Pozisyon açmak için kullanılacak emir tipi (ENUM_ORDER_TYPE sayımının değerlerinden biri).

volume

[in] İstenen pezisyon hacmi.

price

[in] Pozisyon açmak için istenen fiyat.

sl

[in] Stop Loss için tetiklenme fiyatı.

tp

[in] Take Profit tetiklenme fiyatı.

comment=""

[in] Pozisyon yorumu.

Dönüş değeri

Temel yapılar başarıyla denetlenirse 'true', aksi durumda 'false'.

Not

PositionOpen(...) yönteminin başarıyla sonuçlanması her zaman başarılı bir alım-satım işlemi uygulandığı anlamına gelmez. Bu yüzden, alım-satım isteğinin sonucunun (işlem sunucusunun dönüş kodu)ResultRetcode() yöntemi ve ResultDeal() yönteminin dönüş değeri kullanılarak denetlenmesi önemlidir.

PositionModify

Belirtilen sembol üzerindeki açık pozisyonu değiştirir.

```
bool  PositionModify(
   const string  symbol,      // sembol
   double        sl,          // Zarar Durdur fiyatı
   double        tp           // Kar Al fiyatı
   )
```

Pozisyonu belirtilen fiş değerine göre dğiştirir.

```
bool  PositionModify(
   const ulong   ticket,      // pozisyon fişi
   double        sl,          // Zarar Durdur fiyatı
   double        tp           // Kar Al fiyatı
   )
```

Parametreler

symbol

[in] Değiştirilmek istenen pozisyonun sembolünün ismi.

ticket

[in] Değiştirilecek pozisyonun fişi.

sl

[in] Yeni Stop Loss değeri (veya değişim gereksizse eski değer).

tp

[in] Yeni Take Profit değeri (veya değişim gereksizse eski değer).

Dönüş Değeri

Temel yapılar başarıyla denetlenirse 'true', aksi durumda 'false'.

Not

PositionModify(...) yönteminin başarıyla sonuçlanması her zaman başarılı bir alım-satım işlemi uygulandığı anlamına gelmez. Bu yüzden, alım-satım isteğinin sonucunun (işlem sunucusunun dönüş kodu) ResultRetcode() yöntemi kullanılarak denetlenmesi önemlidir.

Netleştirme (netting) sisteminde (ACCOUNT_MARGIN_MODE_RETAIL_NETTING ve ACCOUNT_MARGIN_MODE_EXCHANGE) bir sembol üzerinde sadece bir pozisyon bulunabilir. Bu pozisyon bir veya daha fazla işlemin sonucu açılmış olabilir. Müşteri terminalinde Araçkutusunun "İşlem" sekmesi içinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Çoklu pozisyonlara izin verilemsi durumunda (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) bir sembol üzerinde birden fazla pozisyon açılabilir. Bu durumda PositionModify fonksiyonu en düşük fiş numarasına sahip olan pozisyonu değiştirir.

PositionClose

Belirtilen sembol ile bir açık pozisyonu kapatır.

```
bool  PositionClose(
   const string  symbol,                  // sembol
   ulong         deviation=ULONG_MAX      // sapma
   )
```

Belirtilen fiş değerine sahip olan pozisyonu kapatır.

```
bool  PositionClose(
   const ulong   ticket,                  // pozisyon fişi
   ulong         deviation=ULONG_MAX      // sapma
   )
```

Parametreler

symbol

[in] Kapatılmak istenen pozisyonun sembolünün ismi.

ticket

[in] Kapatılan pozisyonun fişi.

deviation=ULONG_MAX

[in] Mevcut fiyattan maksimum sapma (puan bazında).

Dönüş Değeri

Temel yapılar başarıyla denetlenirse 'true', aksi durumda 'false'.

Not

PositionClose(...) yönteminin başarıyla sonuçlanması her zaman başarılı bir alım-satım işlemi uygulandığı anlamına gelmez. Bu yüzden, alım-satım isteğinin sonucunun (işlem sunucusunun dönüş kodu) ResultRetcode() yöntemi kullanılarak denetlenmesi önemlidir.

Netleştirme (netting) sisteminde (ACCOUNT_MARGIN_MODE_RETAIL_NETTING ve ACCOUNT_MARGIN_MODE_EXCHANGE) bir sembol üzerinde sadece bir pozisyon bulunabilir. Bu pozisyon bir veya daha fazla işlemin sonucu açılmış olabilir. Müşteri terminalinde Araçkutusunun "İşlem" sekmesi içinde birlikte gösterilen mevcut bekleyen emirler ve pozisyonlar birbirleriyle karıştırılmamalıdır.

Çoklu pozisyonlara izin verilemsi durumunda (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING) bir sembol üzerinde birden fazla pozisyon açılabilir. Bu durumda PositionClose fonksiyonu en düşük fiş numarasına sahip olan pozisyonu kapatır.

PositionClosePartial

Hedge'li hesaplama durumunda belirtilen sembol üzerindeki bir açık pozisyonu kısmen kapatır.

```
bool  PositionClosePartial(
   const string  symbol,                  // sembol
   const double  volume,                  // hacim
   ulong         deviation=ULONG_MAX      // sapma
   )
```

Hedge'li hesaplama durumunda belirtilen fiş değerine sahip açık pozisyonu kısmen kapatır.

```
bool  PositionClosePartial(
   const ulong   ticket,                  // pozisyon fişi
   const double  volume,                  // hacim
   ulong         deviation=ULONG_MAX      // sapma
   )
```

Parametreler

symbol

[in] Kısmen kapatılan pozisyonun sembolünün ismi. Eğer pozisyonun kısmen kapatılması için sembol belirtilmişse (fiş değil), bu sembol üzerinde belirtilen SihirliSayıya (MagicNumber - Expert Advisor ID) sahip ilk pozisyon seçilir. Bu yüzden, PositionClosePartial() yöntemini fiş numarası ile kullanmak bazen daha iyidir.

volume

[in] Pozisyondaki, kapatılacak hacim miktarı. Bu değerin pozisyon hacmini geçmesi durumunda pozisyon tamamen kapatılır. Ters yönde pozisyon açılmaz.

ticket

[in] kapatılacak pozisyonun fişi.

deviation=ULONG_MAX

[in] Mevcut fiyattan maksimum sapma (puan bazında).

Dönüş Değeri

Yapıların temel denetimi başarılı ise 'true', aksi durumda 'false'.

Not

PositionClosePartial(...) yönteminin başarıyla sonuçlanması her zaman başarılı bir alım-satım işlemi uygulandığı anlamına gelmez. Bu yüzden, alım-satım isteğinin sonucunun (işlem sunucusunun dönüş kodu) ResultRetcode() yöntemi kullanılarak denetlenmesi önemlidir.

"Netleştirme" sisteminde (ACCOUNT_MARGIN_MODE_RETAIL_NETTING ve ACCOUNT_MARGIN_MODE_EXCHANGE), herhangi bir anda herhangi bir sembol için sadece bir pozisyon açık olabilir (bu pozisyon birkaç işlemin sonucu olabilir). Mevcut bekleyen emirleri Terminalin "AraçKutusu" panelindeki "İşlem" sekmesinde gösterilen pozisyonlarla karıştırmayın.

Pozisyon temsili durumunda (ACCOUNT_MARGIN_MODE_RETAIL_HEDGING), her bir sembol için aynı anda birkaç pozisyon açık olabilir. Bu durumda, PositionClose son fişe sahip olan pozisyonu kapatır.

PositionCloseBy

Pozisyonu belirtilen ters pozisyonun fişini kulanarak kapatır.

```
bool  PositionCloseBy(
   const ulong   ticket,        // pozisyon fişi
   const ulong   ticket_by      // ters pozisyon fişi
   )
```

Parametreler

ticket

[in] Kapatilan pozisyonun fişi.

ticket_by

[in] Kapama için kullanılan ters pozisyonun fişi.

Dönüş değeri

Yapıların denetimi başarılı ise 'true', aksi durumda 'false'.

Not

PositionCloseBy(...) yönteminin başarıyla sonuçlanması her zaman başarılı bir alım-satım işlemi uygulandığı anlamına gelmez. Bu yüzden, alım-satım isteğinin sonucunun (işlem sunucusunun dönüş kodu) ResultRetcode() yöntemi kullanılarak denetlenmesi önemlidir.

Buy

Belirtilen parametrelerle bir uzun pozisyon açar.

```
bool  Buy(
   double         volume,          // pozisyon hacmi
   const string   symbol=NULL,     // sembol
   double         price=0.0,       // fiyat
   double         sl=0.0,          // Stop Loss fiyatı
   double         tp=0.0,          // Take Profit fiyatı
   const string   comment=""       // yorum
   )
```

Parametreler

volume

[in] Pozisyon hacmi.

symbol=NULL

[in] Pozisyon sembolü. Sembol belirtilmemişse geçerli sembol kullanılır.

price=0.0

[in] Fiyat. Fiyat belirtilmemişse geçerli piyasa fiyatı (Ask) kullanılır.

sl=0.0

[in] Stop Loss (zararı durdur) fiyatı.

tp=0.0

[in] Take Profit (kar al) fiyatı.

comment=""

[in] Yorum.

Dönüş değeri

Yapı başarıyla denetlenirse 'true', aksi durumda 'false'.

Not

Buy(...) yönteminin başarıyla sonuçlanması her zaman başarılı bir alım-satım işlemi uygulandığı anlamına gelmez. Bu yüzden, alım-satım isteğinin sonucunun (işlem sunucusunun dönüş kodu) ResultRetcode() yöntemi ve ResultDeal() yönteminin dönüş değeri kullanılarak denetlenmesi önemlidir.

Sell

Belirtilen parametrelerle bir kısa pozisyon açar.

```
bool  Sell(
   double         volume,           // pozisyon hacmi
   const string   symbol=NULL,      // sembol
   double         price=0.0,        // fiyat
   double         sl=0.0,           // Stop Loss fiyatı
   double         tp=0.0,           // Take Profit fiyatı
   const string   comment=""        // yorum
   )
```

Parametreler

volume

[in] Pozisyon hacmi.

symbol=NULL

[in] Pozisyon sembolü. Sembol belirtilmemişse geçerli sembol kullanılır.

price=0.0

[in] Fiyat. Fiyat belirtilmemişse geçerli piyasa fiyatı (Bid) kullanılır.

sl=0.0

[in] Stop Loss (zararı durdur) fiyatı.

tp=0.0

[in] Take Profit (kar al) fiyatı.

comment=""

[in] Yorum.

Dönüş değeri

Yapı başarıyla denetlenirse 'true', aksi durumda 'false'.

Not

Sell(...) yönteminin başarıyla sonuçlanması her zaman başarılı bir alım-satım işlemi uygulandığı anlamına gelmez. Bu yüzden, alım-satım isteğinin sonucunun (işlem sunucusunun dönüş kodu) ResultRetcode() yöntemi ve ResultDeal() yönteminin dönüş değeri kullanılarak denetlenmesi önemlidir.

BuyLimit

Belirtilen parametrelerle Buy Limit tipinde bir bekleyen emir (geçerli piyasa fiyatının daha altından alım emri) yerleştirir.

```
bool  BuyLimit(
   double                volume,                    // emir hacmi
   double                price,                     // emir fiyatı
   const string          symbol=NULL,               // sembol
   double                sl=0.0,                    // Stop Loss fiyatı
   double                tp=0.0,                    // Take Profit fiyatı
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // emir süresi
   datetime              expiration=0,              // emir zaman-aşımı süresi
   const string          comment=""                 // yorum
   )
```

Parametreler

volume

[in] Emir hacmi.

price

[in] Emir fiyatı.

symbol=NULL

[in] Emir sembolü. Sembol belirtilmemişse geçerli sembol kullanılır.

sl=0.0

[in] Stop Loss (zararı durdur) fiyatı.

tp=0.0

[in] Take Profit (kar al) fiyatı.

type_time=ORDER_TIME_GTC

[in] Emir süresi (ENUM_ORDER_TYPE_TIME sayımının değerlerinden biri).

expiration=0

[in] Emir aman-aşımı süresi (sadece, type_time=ORDER_TIME_SPECIFIED ise kullanılır).

comment=""

[in] Emir yorumu.

Dönüş değeri

Yapı başarıyla denetlenirse 'true', aksi durumda 'false'.

Not

BuyLimit(...) yönteminin başarıyla sonuçlanması her zaman başarılı bir alım-satım işlemi uygulandığı anlamına gelmez. Bu yüzden, alım-satım isteğinin sonucunun (işlem sunucusunun dönüş kodu) ResultRetcode() yöntemi ve ResultOrder() yönteminin dönüş değeri kullanılarak denetlenmesi önemlidir.

BuyStop

Belirtilen parametrelerle Buy Stop tipinde bir bekleyen emir (geçerli piyasa fiyatının daha üstünden alım emri) yerleştirir.

```
bool  BuyStop(
   double                volume,                    // emir hacmi
   double                price,                     // emir fiyatı
   const string          symbol=NULL,               // sembol
   double                sl=0.0,                    // Stop Loss fiyatı
   double                tp=0.0,                    // Take Profit fiyatı
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // emir süresi
   datetime              expiration=0,              // emir zaman-aşımı süresi
   const string          comment=""                 // yorum
   )
```

Parametreler

volume

[in] Emir hacmi.

price

[in] Emir fiyatı.

symbol=NULL

[in] Emir sembolü. Sembol belirtilmemişse geçerli sembol kullanılır.

sl=0.0

[in] Stop Loss (zararı durdur) fiyatı.

tp=0.0

[in] Take Profit (kar al) fiyatı.

type_time=ORDER_TIME_GTC

[in] Emir süresi (ENUM_ORDER_TYPE_TIME sayımının değerlerinden biri).

expiration=0

[in] Emir zaman-aşımı süresi (sadece, type_time=ORDER_TIME_SPECIFIED ise kullanılır).

comment=""

[in] Emir yorumu.

Dönüş değeri

Yapı başarıyla denetlenirse 'true', aksi durumda 'false'.

Not

BuyStop(...) yönteminin başarıyla sonuçlanması her zaman başarılı bir alım-satım işlemi uygulandığı anlamına gelmez. Bu yüzden, alım-satım isteğinin sonucunun (işlem sunucusunun dönüş kodu) ResultRetcode() yöntemi ve ResultOrder() yönteminin dönüş değeri kullanılarak denetlenmesi önemlidir.

SellLimit

Belirtilen parametrelerle Sell Limit tipinde bir bekleyen emir (geçerli piyasa fiyatının daha üstünden satış emri) yerleştirir.

```
bool  SellLimit(
   double                volume,                    // emir hacmi
   double                price,                     // emir fiyatı
   const string          symbol=NULL,               // sembol
   double                sl=0.0,                    // Stop Loss fiyatı
   double                tp=0.0,                    // Take Profit fiyatı
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // emir süresi
   datetime              expiration=0,              // emir zaman-aşımı süresi
   const string          comment=""                 // yorum
   )
```

Parametreler

volume

[in] Emir hacmi.

price

[in] Emir fiyatı.

symbol=NULL

[in] Emir sembolü. Sembol belirtilmemişse geçerli sembol kullanılır.

sl=0.0

[in] Stop Loss (zararı durdur) fiyatı.

tp=0.0

[in] Take Profit (kar al) fiyatı.

type_time=ORDER_TIME_GTC

[in] Emir süresi (ENUM_ORDER_TYPE_TIME sayımının değerlerinden biri).

expiration=0

[in] Emir zaman-aşımı süresi (sadece, type_time=ORDER_TIME_SPECIFIED ise kullanılır).

comment=""

[in] Emir yorumu.

Dönüş değeri

Yapı başarıyla denetlenirse 'true', aksi durumda 'false'.

Not

SellLimit(...) yönteminin başarıyla sonuçlanması her zaman başarılı bir alım-satım işlemi uygulandığı anlamına gelmez. Bu yüzden, alım-satım isteğinin sonucunun (işlem sunucusunun dönüş kodu) ResultRetcode() yöntemi ve ResultOrder() yönteminin dönüş değeri kullanılarak denetlenmesi önemlidir.

SellStop

Belirtilen parametrelerle Sell Stop tipinde bir bekleyen emir (geçerli piyasa fiyatının daha altından satış emri) yerleştirir.

```
bool  SellStop(
   double                volume,                    // emir hacmi
   double                price,                     // emir fiyatı
   const string          symbol=NULL,               // sembol
   double                sl=0.0,                    // Stop Loss fiyatı
   double                tp=0.0,                    // Take Profit fiyatı
   ENUM_ORDER_TYPE_TIME  type_time=ORDER_TIME_GTC,  // emir süresi
   datetime              expiration=0,              // emir zaman-aşımı süresi
   const string          comment=""                 // yorum
   )
```

Parametreler

volume

[in] Emir hacmi.

price

[in] Emir fiyatı.

symbol=NULL

[in] Emir sembolü. Sembol belirtilmemişse geçerli sembol kullanılır.

sl=0.0

[in] Stop Loss (zararı durdur) fiyatı.

tp=0.0

[in] Take Profit (kar al) fiyatı.

type_time=ORDER_TIME_GTC

[in] Emir süresi (ENUM_ORDER_TYPE_TIME sayımının değerlerinden biri).

expiration=0

[in] Emir zaman-aşımı süresi (sadece, type_time=ORDER_TIME_SPECIFIED ise kullanılır).

comment=""

[in] Emir yorumu.

Dönüş değeri

Yapı başarıyla denetlenirse 'true', aksi durumda 'false'.

Not

SellStop(...) yönteminin başarıyla sonuçlanması her zaman başarılı bir alım-satım işlemi uygulandığı anlamına gelmez. Bu yüzden, alım-satım isteğinin sonucunun (işlem sunucusunun dönüş kodu) ResultRetcode() yöntemi ve ResultOrder() yönteminin dönüş değeri kullanılarak denetlenmesi önemlidir.

Request

Son istek yapısının kopyasını alır.

```
void  Request(
   MqlTradeRequest&  request      // hedef yapı
   ) const
```

Parametreler

request

[out] MqlTradeRequest tipli yapının referansı.

Dönüş değeri

Yok.

RequestAction

Alım-satım işlem tipini alır.

```
ENUM_TRADE_REQUEST_ACTIONS  RequestAction() const
```

Dönüş değeri

Son istekte kullanılan alım-satım işleminin tipi.

RequestActionDescription

Alım-satım işlem tipini dizgi biçiminde alır.

```
string  RequestActionDescription() const
```

Dönüş değeri

Son istekte kullanılan alım-satım işleminin tipi (dizgi biçiminde).

RequestMagic

Uzman Danışman tanımlayıcısını (magic number) alır.

```
ulong  RequestMagic() const
```

Dönüş değeri

Son istekte kullanılmış olan Uzman Danışmanın tanımlayıcısı.

RequestOrder

Son istekte kullanılmış olan emir fişini alır.

```
ulong  RequestOrder() const
```

Dönüş değeri

Son istekte kullanılmış olan emir fişi.

RequestSymbol

Son istekte kullanılmış olan sembolün ismini alır

```
string  RequestSymbol() const
```

Dönüş değeri

Son istekte kullanılmış olan sembolün ismi.

RequestVolume

Son istekte kullanılmış olan işlem hacmini lot cinsinden alır.

```
double  RequestVolume() const
```

Dönüş değeri

Son istekte kullanılmış olan işlem hacmi (lot cinsinden).

RequestPrice

Son istekte kullanılmış olan fiyatı alır.

```
double  RequestPrice() const
```

Dönüş değeri

Son istekte kullanılmış olan emir fiyatı.

RequestStopLimit

Son istekte kullanılmış olan Stop Limit tipli bekleyen emrin fiyatını alır.

```
double  RequestStoplimit() const
```

Dönüş değeri

Son istekte kullanılmış olan Stop Limit tipli bekleyen emrin fiyatı.

RequestSL

Son istekte kullanılmış olan Stop Loss fiyatını alır.

```
double  RequestSL() const
```

Dönüş değeri

Son istekte kullanılmış olan Stop Loss fiyatı.

RequestTP

Son istekte kullanılmış olan Take Profit fiyatını alır.

```
double  RequestTP() const
```

Dönüş değeri

Son istekte kullanılmış olan Take Profit fiyatı.

RequestDeviation

Son istekte kullanılmış olan fiyat sapması değerini alır.

```
ulong  RequestDeviation() const
```

Dönüş değeri

Son istekte kullanılmış olan fiyat sapması değeri.

RequestType

Son istekte kullanılan emir tipini alır.

```
ENUM_ORDER_TYPE  RequestType() const
```

Dönüş değeri

Son istekte kullanılan emir tipi (ENUM_ORDER_TYPE sayımının değerlerinden biri).

RequestTypeDescription

Son istekte kullanılmış olan emir tipini dizgi biçiminde alır.

```
string  RequestTypeDescription() const
```

Dönüş değeri

Son istekte kullanılmış olan emir tipi (dizgi biçiminde).

RequestTypeFilling

Son istekte kullanılmış olan emir karşılama türü alır.

```
ENUM_ORDER_TYPE_FILLING  RequestTypeFilling() const
```

Dönüş değeri

Son istekte kullanılmış olan emir karşılama türü (ENUM_ORDER_TYPE_FILLING sayımının değerlerinden biri).

RequestTypeFillingDescription

Son istekte kullanılmış olan emir karşılama türünü dizgi biçiminde alır.

```
string  RequestTypeFillingDescription() const
```

Dönüş değeri

Son istekte kullanılmış olan emir karşılama türü (dizgi biçiminde).

RequestTypeTime

Son istekte kullanılmış olan emir geçerlilik periyodunu alır.

```
ENUM_ORDER_TYPE_TIME  RequestTypeTime() const
```

Dönüş değeri

Son istekte kullanılmış olan emrin geçerlilik periyodu (ENUM_ORDER_TYPE_TIME sayımının değerlerinden biri).

RequestTypeTimeDescription

Son istekte kullanılmış olan emir geçerlilik periyodunu dizgi biçiminde alır.

```
string  RequestTypeTimeDescription() const
```

Dönüş değeri

Son istekte kullanılmış olan emir geçerlilik periyodu (dizgi biçiminde).

RequestExpiration

Son istekte kullanılmış olan zaman-aşımı süresinin değerini alır.

```
datetime  RequestExpiration() const
```

Dönüş değeri

Son istekte kullanılmış olan zaman-aşımı süresi.

RequestComment

Son istekte kullanılmış olan emir yorumunu alır.

```
string  RequestComment() const
```

Dönüş değeri

Son istekte kullanılmış olan emir yorumu.

RequestPosition

Pozisyonun fişini alır.

```
ulong  RequestPosition() const
```

Dönüş değeri

Son istekte kullanılan pozisyonun fişi.

RequestPositionBy

Ters pozisyonun fişini alır.

```
ulong  RequestPositionBy() const
```

Dönüş değeri

Son istekte kullanılan ters pozisyonun fişi.

Result

Son istek sonucu yapısının kopyasını alır.

```
void  Result(
   MqlTradeResult&  result      // referans
   ) const
```

Parametreler

result

[out] MqlTradeResult tipli yapının referansı.

Dönüş değeri

Yok.

ResultRetcode

İstek sonucunun kodunu alır.

```
uint  ResultRetcode() const
```

Dönüş değeri

İstek sonucunun Kodu.

ResultRetcodeDescription

İstek sonucunun kodunu dizgi biçiminde alır.

```
string  ResultRetcodeDescription() const
```

Dönüş değeri

Son istek sonucunun kodu (dizgi biçiminde).

ResultDeal

İşlemin fişini alır.

```
ulong  ResultDeal() const
```

Dönüş değeri

İşlem fişi (işlem gerçekleşmişse).

ResultOrder

Emir fişini alır.

```
ulong  ResultOrder() const
```

Dönüş değeri

Emir fişi (emir yerleştirilmişse).

ResultVolume

Emir veya işlem hacmini alır.

```
double  ResultVolume() const
```

Dönüş değeri

Emrin veya işlemin hacmi.

ResultPrice

Broker tarafından onaylanan fiyatı alır.

```
double  ResultPrice() const
```

Dönüş değeri

Broker tarafından onaylanan fiyat.

ResultBid

Geçerli teklif fiyatını alır (yeniden teklif).

```
double  ResultBid() const
```

Dönüş değeri

Geçerli teklif fiyatı (yeniden teklif).

ResultAsk

Geçerli istek (ask) fiyatını alır (yeniden teklif).

```
double  ResultAsk() const
```

Dönüş değeri

Geçerli istek (ask) fiyatı (yeniden teklif).

ResultComment

Broker yorumunu alır.

```
string  ResultComment() const
```

Dönüş değeri

İşlem için broker yorumu.

CheckResult

Son isteğin denetim sonucu yapısının kopyasını alır.

```
void  CheckResult(
   MqlTradeCheckResult&  check_result      // referans
   ) const
```

Parametreler

check_result

[out] MqlTradeCheckResult tipli hedef yapısının referansı.

Dönüş değeri

Yok.

CheckResultRetcode

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'retcode' alanının değerini alır.

```
uint  CheckResultRetcode() const
```

Dönüş değeri

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'retcode' alanının değeri (hata kodu).

CheckResultRetcodeDescription

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'retcode' (dönüş kodu) alanının dizgi biçimli açılamasını alır.

```
string  ResultRetcodeDescription() const
```

Dönüş değeri

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'retcode' alanının dizgi biçimli açılaması (hata kodu).

CheckResultBalance

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'balance' (bakiye) alanının değerini alır.

```
double  CheckResultBalance() const
```

Dönüş değeri

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'balance' alanının değeri (alım-satım işleminin ardından oluşan bakiye değeri).

CheckResultEquity

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'equity' (varlık) alanının değerini alır.

```
double  CheckResultEquity() const
```

Dönüş değeri

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'equity' alanının değeri (alım-satım işleminin ardından oluşan varlık değeri).

CheckResultProfit

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'profit' (kar) alanının değerini alır.

```
double  CheckResultProfit() const
```

Dönüş değeri

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'profit' alanının değeri (alım-satım işleminin ardından oluşacak olan kar değeri).

CheckResultMargin

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'margin' (teminat) alanının değerini alır.

```
double  CheckResultMargin() const
```

Dönüş değeri

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'margin' alanının değeri (alım-satım işlemi için gereken teminat değeri).

CheckResultMarginFree

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'margin_free' (serbest teminat) alanının değerini alır.

```
double  CheckResultMarginFree() const
```

Dönüş değeri

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'margin_free' alanının değeri (alım-satım işleminin ardından oluşan serbest teminat değeri).

CheckResultMarginLevel

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'margin_level' (teminat seviyesi) alanının değerini alır.

```
double  CheckResultMarginLevel() const
```

Dönüş değeri

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'margin_level' alanının değeri (alım-satım işleminin ardından ayarlanacak olan teminat seviyesi).

CheckResultComment

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'comment' (yorum) alanının değerini alır.

```
string  CheckResultComment() const
```

Dönüş değeri

İstek geçerliliğinin denetimi sırasında doldurulmuş olan MqlTradeCheckResult yapısının 'comment' alanının değeri (cevap kodunun yorumu, hata açıklaması).

PrintRequest

Son istek parametrelerini bültene yazdırır.

```
void  PrintRequest() const
```

Dönüş değeri

Yok.

PrintResult

Son istek sonuçlarını bültene yazdırır.

```
void  PrintResult() const
```

Dönüş değeri

Yok.

FormatRequest

Son istek parametreleriye biçimlendirilmiş bir dizgi hazırlar.

```
string  FormatRequest(
   string&                 str,          // hedef dizgi
   const MqlTradeRequest&  request       // istek
   ) const
```

Parametreler

str

[in] Referansla geçirilen hedef dizgi.

request

[in] Son istek parametrelerini içeren MqlTradeRequest tipli bir yapı.

Dönüş değeri

Yok.

FormatRequestResult

Son isteğin uygulama sonuçlarıyla biçimlendirilmiş bir dizgi hazırlar

```
string  FormatRequestResult(
   string&                 str,          // dizgi
   const MqlTradeRequest&  request,      // istek yapısı
   const MqlTradeResult&   result        // sonuç yapısı
   ) const
```

Parametreler

str

[in] Referansla geçirilen hedef dizgi.

request

[in] Son istek parametrelerini içeren MqlTradeRequest tipli bir yapı.

result

[in] Son istek parametrelerini içeren MqlTradeResult tipli bir yapı.

Dönüş değeri

Yok.

CTerminalInfo

CTerminalInfo sınıfı, mql5 program ortamının özelliklerine kolay erişim sağlamak için tasarlanmıştır.

Açıklama

CTerminalInfo sınıfı, mql5 program ortamının özelliklerine erişim sağlar.

Bildirim

```
class CTerminalInfo : public CObject
```

Başlık

```
#include <Trade\TerminalInfo.mqh>
```

Kalıtım hiyerarşisi

CObject

CTerminalInfo

Gruplarına göre sınıf yöntemleri

Tamsayı tipli özelliklere erişmek için kullanılan yöntemler	
Build	Müşteri terminalinin kurulum numarasını alır
IsConnected	Alım-satım sunucusuyla kurulan bağlantı hakkında bilgi alır
IsDLLsAllowed	DLL kullanım izni hakkında bilgi alır
IsTradeAllowed	Alım-satım işlemi izni hakkında bilgi alır
IsEmailEnabled	Terminal ayarlarında belirtilen "STMP-sunucusuna e-posta gönderme" ve "oturum-açma" izinleri hakkında bilgi alır
IsFtpEnabled	Terminal ayarlarında belirtilen "FTP-sunucusuna rapor gönderme" ve "oturum-açma" izinleri hakkında bilgi alır
MaxBars	Çizelgede kullanılan maksimum çubuk sayısı hakkında bilgi alır
CodePage	Müşteri terminalindeki dilin kod sayfası hakkında bilgi alır
CPUCores	CPU çekirdekleri hakkında bilgi alır
MemoryPhysical	Fiziksel bellek hakkında (Mb cinsinden) bilgi alır
MemoryTotal	Terminal/temsilci işlemleri için ayrılmış toplam bellek miktarı hakkında (Mb cinsinden) bilgi alır.
MemoryAvailable	Terminal/temsilci işlemleri için kullanılabilir olan serbest bellek miktarı hakkında (Mb cinsinden) bilgi alır.

Tamsayı tipli özelliklere erişmek için kullanılan yöntemler	
MemoryUsed	Terminal/temsilci işlemleri tarafından kullanılan bellek miktarı hakkında (Mb cinsinden) bilgi alır.
IsX64	Müşteri terminalinin tipi hakkında (32/64 bit) bilgi alır
OpenCLSupport	Grafik kartı tarafından desteklenen OpenCL sürümü hakkında bilgi alır
DiskSpace	Serbest disk alanı hakkında (Mb cinsinden) bilgi alır
string tipli özelliklere erişmek için kullanılan yöntemler	
Language	Müşteri terminalinin dili hakkında bilgi alır
Name	Müşteri terminalinin ismini alır
Company	müşteri terminalinin şirket ismini alır
Path	Müşteri terminali klasörünü alır
DataPath	Müşteri terminalinin veri klasörünün adresini alır
CommonDataPath	Bilgisayara yüklenmiş tüm müşteri terminalleri için kullanılan ortak veri klasörünün adresini alır
MQL5 API fonksiyonlarına erişim	
InfoInteger	Tamsayı tipli parametrenin değerini alır
InfoString	string tipli parametrenin değerini alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Build

Müşteri terminalinin kurulum numarasını alır.

```
int  CBuild() const
```

Dönüş değeri

Müşteri terminalinin kurulum numarası.

Not

Kurulum numarasını almak için TerminalInfoInteger() fonksiyonunu kullanır (TERMINAL_BUILD özelliği).

IsConnected

Alım-satım sunucusuyla kurulan bağlantı hakkında bilgi alır.

```
bool  IsConnected() const
```

Dönüş değeri

Terminal alım-satım sunucusuna bağlıysa 'true', aksi durumda 'false'.

Not

Bağlantı bilgisini almak için TerminalInfoInteger() fonksiyonunu kullanır (TERMINAL_CONNECTED özelliği).

IsDLLsAllowed

DLL kullanım izni hakkında bilgi alır.

```
bool  IsDLLsAllowed() const
```

Dönüş değeri

DLL kullanımına izin verilmişse 'true', aksi durumda 'false'.

Not

DLL kulanım izni bilgisini TerminalInfoInteger() fonksiyonunu kullanarak alır (TERMINAL_DLLS_ALLOWED özelliği).

IsTradeAllowed

Alım-satım işlemi izni hakkında bilgi alır.

```
bool  IsTradeAllowed() const
```

Dönüş değeri

Alım-satım işlemine izin verilmişse 'true', aksi durumda 'false'.

Not

Alım-satım işlemi izni hakkında bilgi almak için TerminalInfoInteger() fonksiyonu (TERMINAL_TRADE_ALLOWED özelliği).

IsEmailEnabled

Terminal ayarlarında belirtilen "STMP-sunucusuna e-posta gönderme" ve "oturum-açma" izinleri hakkında bilgi alır

```
bool  IsEmailEnabled() const
```

Dönüş değeri

E-posta gönderme eylemine izin veriliyorsa 'true', aksi durumda 'false'.

Not

E-posta izni hakkında bilgi almak için TerminalInfoInteger() fonksiyonunu kullanır (TERMINAL_EMAIL_ENABLED özelliği).

IsFtpEnabled

Terminal ayarlarında belirtilen "FTP-sunucusuna rapor gönderme" ve "oturum-açma" izinleri hakkında bilgi alır

```
bool  IsFtpEnabled() const
```

Dönüş değeri

FTP sunucusuna rapor gönderme eylemine izin veriliyorsa 'true', aksi durumda 'false'.

Not

FTP-sunucusuna rapor gönderme izni hakkında bilgi almak için TerminalInfoInteger() fonksiyonunu kullanır (TERMINAL_FTP_ENABLED özelliği).

MaxBars

Müşteri terminalinin ayarlarında belirtilen, çizelge üzerindeki maksimum çubuk sayısını alır.

```
int  MaxBars() const
```

Dönüş değeri

Çizelge üzerindeki maksimum çubuk sayısı.

Not

Çizelge üzerindeki maksimum çubuk sayısını almak için TerminalInfoInteger() fonksiyonunu kullanır (TERMINAL_MAXBARS özelliği).

CodePage

Müşteri terminalindeki dilin kod sayfası hakkında bilgi alır.

```
int  CodePage() const
```

Dönüş değeri

Müşteri terminalindeki dilin kod sayfası.

Not

Kod sayfasını almak için TerminalInfoInteger() fonksiyonunu kullanır (TERMINAL_CODEPAGE özelliği).

CPUCores

CPU çekirdeklerinin sayısı hakkında bilgi alır.

```
int  CPUCores() const
```

Dönüş değeri

CPU çekirdeklerinin sayısı.

Not

CPU çekirdeklerinin sayısını almak için TerminalInfoInteger() fonksiyonunu kullanır (TERMINAL_CPU_CORES özelliği).

MemoryPhysical

Fiziksel bellek hakkında (Mb cinsinden) bilgi alır

```
int  MemoryPhysical() const
```

Dönüş değeri

Fiziksel bellek (Mb cinsinden).

Not

Fiziksel bellek alanı hakkında bilgi almak için TerminalInfoInteger() fonksiyonunu kullanır (TERMINAL_MEMORY_PHYSICAL özelliği).

MemoryTotal

Terminal/temsilci işlemleri için ayrılmış toplam bellek miktarı hakkında (Mb cinsinden) bilgi alır.

```
int  MemoryTotal() const
```

Dönüş değeri

Terminal/temsilci işlemleri için ayrılmış toplam bellek miktarı.

Not

Tooplam bellek miktarı hakkında bilgi almak için TerminalInfoInteger() fonksiyonunu kullanır (TERMINAL_MEMORY_TOTAL özelliği).

MemoryAvailable

Müşteri terminali ve temsilciler için ayrılan serbest bellek alanı hakkında (Mb cinsinden) bilgi alır

```
int  MemoryTotal() const
```

Dönüş değeri

Terminal ve temsilciler için ayrılan serbest bellek alanı.

Not

Serbest bellek alanı hakkında bilgi almak için TerminalInfoInteger() fonksiyonunu kullanır (TERMINAL_MEMORY_TOTAL özelliği).

MemoryUsed

Terminal/temsilci işlemleri tarafından kullanılan bellek miktarı hakkında (Mb cinsinden) bilgi alır.

```
int  MemoryUsed() const
```

Dönüş değeri

Terminal/temsilci işlemleri tarafından kullanılan bellek miktarı (Mb cinsinden).

Not

Kullanılan bellek miktarı hakkında bilgi almak için TerminalInfoInteger() fonksiyonunu kullanır (TERMINAL_MEMORY_USED özelliği).

IsX64

Müşteri terminalinin tipi hakkında bilgi alır

```
bool  IsX64() const
```

Dönüş değeri

64-bit sürümü ise 'true', aksi durumda 'false'.

Not

Müşteri terminalinin tipi hakkında bilgi almak için TerminalInfoInteger() fonksiyonunu kullanır (TERMINAL_X64 özelliği).

OpenCLSupport

Grafik kartı tarafından desteklenen OpenCL sürümü hakkında bilgi alır.

```
int  OpenCLSupport() const
```

Dönüş değeri

Dönüş değeri şu şekildedir: 0x00010002 = "1.2". 0 değeri, OpenCL desteğinin bulunmadığını gösterir.

Not

OpenCL sürümü hakkında bilgi almak için TerminalInfoInteger() fonksiyonunu kullanır (TERMINAL_OPENCL_SUPPORT özelliği).

DiskSpace

Müşteri terminali ve temsilciler için ayrılan serbest disk alanı hakkında (Mb cinsinden) bilgi alır

```
int  MDiskSpace() const
```

Dönüş değeri

Müşteri terminali ve temsilciler için ayrılan serbest disk alanı (dosyalar için MQL5\Files klasörüne kaydedilir).

Not

Serbest disk alanı bilgisini almak için TerminalInfoInteger() fonksiyonunu kullanır (TERMINAL_DISK_SPACE özelliği).

Language

Müşteri terminalinin dili hakkında bilgi alır.

```
string  Language() const
```

Dönüş değeri

Müşteri terminalinde kullanılan dil.

Not

Dil hakkında bilgi almak için TerminalInfoString() fonksiyonunu kullanır (TERMINAL_LANGUAGE özelliği).

Name

Gets the information of the name of the client terminal.

```
string  Name() const
```

Dönüş değeri

Name of the client terminal.

Not

To get the name of the client terminal it uses the TerminalInfoString() function (TERMINAL_NAME property).

Company

Brokerın ismi hakkında bilgi alır.

```
string  Company() const
```

Dönüş değeri

Brokerın ismi.

Not

Broker ismini almak için TerminalInfoString() fonksiyonunu kullanır (TERMINAL_COMPANY özelliği).

Path

Terminal veri klasörü hakkında bilgi alır.

```
string  Path() const
```

Dönüş değeri

Terminal veri klasörü.

Not

Terminal veri klasörünün bilgisini almak için TerminalInfoString() fonksiyonunu kullanır (TERMINAL_PATH özelliği).

DataPath

Terminal veri klasörü hakkında bilgi alır.

```
string  DataPath() const
```

Dönüş değeri

Müşteri terminalinin veri klasörü.

Not

Müşteri terminalinin veri klasörünü almak için TerminalInfoString() fonksiyonunu kullanır (TERMINAL_DATA_PATH özelliği).

CommonDataPath

Bilgisayara yüklenmiş tüm müşteri terminalleri için kullanılan ortak veri klasörünün adresini alır

```
string  CommonDataPath() const
```

Dönüş değeri

Ortak veri klasörü.

Not

Ortak klasörü almak için TerminalInfoString() fonksiyonunu kullanır (COMMON_DATA_PATH özelliği).

InfoInteger

Mql5 program ortamının karşılık gelen özelliğinin değerini alır.

```
int  TerminalInfoInteger(
   int  property_id      // özellik tanımlayıcısı
   );
```

Parametreler

property_id

[in] Özellik tanımlayıcısı. ENUM_TERMINAL_INFO_INTEGER sayımının değerlerinden biri olabilir.

Dönüş değeri

int tipli bir değer.

Not

Özellik değerini almak için TerminalInfoInteger() fonksiyonunu kullanır.

InfoString

Mql5 program ortamının karşılık gelen özelliğinin değerini alır. Özellik string tipinde olmalıdır.

```
string  TerminalInfoString(
   int  property_id      // özellik tanımlayıcısı
   );
```

Parametreler

property_id

[in] Özellik tanımlayıcısı. ENUM_TERMINAL_INFO_STRING sayımının değerlerinden biri olabilir.

Dönüş değeri

string tipli değer.

Not

Özellik değerini almak için TerminalInfoString() fonksiyonunu kullanır.

Alım-Satım Stratejisi Sınıfları

Bu bölüm, alım-satım stratejilerinin oluşturulması ve sınanması için geliştirilen sınıflarla çalışmanın detaylarını ve MQL5 standart kütüphanesinin ilgili bileşenlerinin açıklamalarını içermektedir.

Bu sınıfların kullanımı alım-satım stratejileri geliştirirken programcıya zaman kazandıracaktır.

MQL5 Standart Kütüphanesi (alım-satım stratejileri açısından) terminalin çalışma dizininde Include\Expert klasöründe yer almaktadır.

Temel sınıflar	Açıklama
CExpertBase	Alım-satım stratejisi sınıfları için temel sınıf
CExpert	Uzman danışman için temel sınıf
CExpertSignal	Alım-satım sinyalleri için temel sınıf
CExpertTrailing	Trailing (iz-süren) Stop için temel sınıf
CExpertMoney	Para yönetimi için temel sınıf

Alım-Satım Sinyali Sınıfları	Açıklama
CSignalAC	Accelerator Oscillator göstergesinin piyasa modellerini temel alan sinyallerin modülü.
CSignalAMA	Adaptive Moving Average göstergesinin piyasa modellerini temel alan sinyallerin modülü.
CSignalAO	Awesome Oscillator göstergesinin piyasa modellerini temel alan sinyallerin modülü.
CSignalBearsPower	Bears Power osilatörünün piyasa modellerini temel alan sinyallerin modülü.
CSignalBullsPower	Bulls Power osilatörünün piyasa modellerini temel alan sinyallerin modülü.
CSignalCCI	Commodity Channel Index osilatörünün piyasa modellerini temel alan sinyallerin modülü.
CSignalDeM	DeMarker osilatörünün piyasa modellerini temel alan sinyallerin modülü.
CSignalDEMA	Double Exponential Moving Average göstergesinin piyasa modellerini temel alan sinyallerin modülü.
CSignalEnvelopes	Envelopes göstergesinin piyasa modellerini temel alan sinyallerin modülü.
CSignalFrAMA	Fractal Adaptive Moving Average göstergesinin piyasa modellerini temel alan sinyallerin modülü.
CSignalITF	Sinyallerin zaman ile filtreleme modülü.

Alım-Satım Sinyali Sınıfları	Açıklama
CSignalMACD	MACD osilatörünün piyasa modellerini temel alan sinyallerin modülü.
CSignalMA	Moving Average göstergesinin piyasa modellerini temel alan sinyallerin modülü.
CSignalSAR	Parabolic SAR göstergesinin piyasa modellerini temel alan sinyallerin modülü.
CSignalRSI	Relative Strength Index osilatörünün piyasa modellerini temel alan sinyallerin modülü.
CSignalRVI	Relative Vigor Index osilatörünün piyasa modellerini temel alan sinyallerin modülü.
CSignalStoch	Stochastic osilatörün piyasa modellerini temel alan sinyallerin modülü.
CSignalTRIX	Triple Exponential Average göstergesinin piyasa modellerini temel alan sinyallerin modülü.
CSignalTEMA	Triple Exponential Moving Average göstergesinin piyasa modellerini temel alan sinyallerin modülü.
CSignalWPR	Williams Percent Range osilatörünün piyasa modellerini temel alan sinyallerin modülü.

Trailing Stop Sınıfları	Açıklama
CTrailingFixedPips	Bu sınıf, sabit puanlara dayanan Trailing (iz-süren) Stop algoritması uygular
CTrailingMA	Bu sınıf, hareketli ortalama göstergesinin değerlerine dayanan Trailing Stop algoritması uygular
CTrailingNone	Bir saplama (stub) sınıf, herhangi bir Trailing Stop algoritması uygulamaz
CTrailingPSAR	Bu sınıf, Parabolic SAR göstergesinin değerlerine dayanan Trailing Stop algoritması uygular

Para Yönetimi Sınıfları	Açıklama
CMoneyFixedLot	Önceden belirlenmiş sabit lot değeri ile işlem yapmak için düzenlenmiş bir sınıf.
CMoneyFixedMargin	Önceden belirlenmiş sabit teminat değeri ile işlem yapmak için düzenlenmiş bir sınıf.
CMoneyFixedRisk	Önceden belirlenmiş sabit risk değeri ile işlem yapmak için düzenlenmiş bir sınıf.
CMoneyNone	İzin verilen en küçük lot değeri ile işlem yapmak için düzenlenmiş bir sınıf.

Para Yönetimi Sınıfları	Açıklama
CMoneySizeOptimized	Önceki işlemlerin sonuçlarına göre belirlenen değişken lot değeri ile işlem yapmak için düzenlenmiş bir sınıf.

Uzman Danışmanlar için temel sınıf

Bu bölüm, alım-satım stratejilerinin oluşturulması ve sınanması için geliştirilen sınıflarla çalışmanın detaylarını ve MQL5 standart kütüphanesinin ilgili bileşenlerinin açıklamalarını içermektedir.

Bu sınıfların kullanımı alım-satım stratejileri geliştirirken programcıya zaman kazandıracaktır.

MQL5 Standart Kütüphanesi (alım-satım stratejileri açısından) terminalin çalışma dizininde Include\Expert klasöründe yer almaktadır.

Sınıf	Açıklama
CExpertBase	Alım-satım stratejisi sınıfları için temel sınıf
CExpert	Uzman danışman için temel sınıf
CExpertSignal	Alım-satım sinyalleri için temel sınıf
CExpertTrailing	Trailing (iz-süren) Stop için temel sınıf
CExpertMoney	Para yönetimi için temel sınıf

CExpertBase

CExpertBase sınıfı, CExpert sınıfının ve tüm alım-satım stratejilerinin temel sınıfıdır.

Açıklama

CExpertBase, tüm Uzman Danışmanlarda yaygın olarak kullanılan veri ve yöntemleri sağlar.

Bildirim

```
class CExpertBase : public CObject
```

Başlık

```
#include <Expert\ExpertBase.mqh>
```

Kalıtım hiyerarşisi

CObject

CExpertBase

İlk nesil

CExpert, CExpertMoney, CExpertSignal, CExpertTrailing

Sınıf Yöntemleri

Ortak yöntemler:

Başlatma	
virtual Init	Sınıf örneği başlatma yöntemi
virtual ValidationSettings	Ayarları denetler
Parametreler	
Symbol	Sembolü ayarlar
Period	Zaman-dilimini ayarlar
Magic	Uzman Danışmanın tanımlayıcısını ayarlar
Göstergeler ve zaman serileri	
virtual SetPriceSeries	Dışsal zaman-serileri (fiyat serileri) için işaretçi ayarlar
virtual SetOtherSeries	Dışsal zaman-serileri (fiyat haricindeki seriler) için işaretçi ayarlar
virtual InitIndicators	Göstergeleri ve zaman-serilerini başlatır
Korunan Veriye Erişim	
InitPhase	Nesne başlatımının mevcut aşamasını alır

Başlatma	
TrendType	Trend tipini ayarlar
UsedSeries	Kullanılan zaman-serilerinin bit-maskesini alır
EveryTick	"Every tick" (tüm tikler) bayrağını ayarlar
Zaman-serilerine erişim	
Open	Open (açılış fiyatı) serisinin elemanını indis kullanarak alır
High	High (yüksek fiyat) serisinin elemanını indis kullanarak alır
Low	Low (düşük fiyat) serisinin elemanını indis kullanarak alır
Close	Close (kapanış fiyatı) serisinin elemanını indis kullanarak alır
Spread	Spread (alış-satış farkı) serisinin elemanını indis kullanarak alır
Time	Time (zaman) serisinin elemanını indis kullanarak alır
TickVolume	TickVolume (tik hacmi) serisinin elemanını indis kullanarak alır
RealVolume	RealVolume (işlem hacmi) serisinin elemanını indis kullanarak alır

Korunan yöntemler:

Zaman-serilerinin başlatılması	
InitOpen	Open serisinin başlatma yöntemi
InitHigh	High serisinin başlatma yöntemi
InitLow	Low serisinin başlatma yöntemi
InitClose	Close serisinin başlatma yöntemi
InitSpread	Spread serisinin başlatma yöntemi
InitTime	Time serisinin başlatma yöntemi
InitTickVolume	TickVolume serisinin başlatma yöntemi
InitRealVolume	RealVolume serisinin başlatma yöntemi
Servis Yöntemleri	
virtual PriceLevelUnit	Fiyat seviye birimini alır
virtual StartIndex	Analiz edilecek başlangıç çubuğunun indisini alır
virtual CompareMagic	Uzman danışmanın tanıtıcı değerini belirtilen değerle karşılaştırır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

InitPhase

Nesne başlatımının mevcut aşamasını alır.

```
ENUM_INIT_PHASE  InitPhase()
```

Dönüş değeri

Nesne başlatımının mevcut aşaması.

Not

Nesne başlatımı çeşitli aşamalar içerir:

1. Başlatma aşaması.

- başlatım - yapıcının tamamlanmasının ardından
- sonlandırma - Init(...) yönteminin başarıyla tamamlanmasının ardından.
- izin verilen - Init(...) yönteminin çağrısı
- izin verilmeyen - ValidationSettings() ve diğer başlatma töntemlerinin çağrıları

2. Parametrelerin ayarlanması aşaması. Bu aşamada, göstergelerin oluşturulması için kullanılan nesnelerin parametrelerini ayarlamanız gerekir.

- başlatım - Init(...) yönteminin başarıyla tamamlanmasının ardından
- sonlandırma - ValidationSettings() yönteminin başarıyla tamamlanmasının ardından
- izin verilen - Symbol(...) ve Period(...) yöntemlerinin çağrıları
- izin verilmeyen - Init(...), SetPriceSeries(...), SetOtherSeries(...) ve InitIndicators(...) yöntemlerinin çağrıları

3. Parametrelerin kontrolü.

- başlatım - ValidationSettings() yönteminin başarıyla tamamlanmasının ardından
- sonlandırma - InitIndicators(...) yönteminin başarıyla tamamlanmasının ardından
- izin verilen - Symbol(...), Period(...) ve InitIndicators(...) yöntemlerinin çağrıları
- izin verilmeyen - tüm diğer başlatma yöntemleri

4. Başlatmanın son aşaması.

- başlatım - InitIndicators(...) yönteminin başarıyla tamamlanmasının ardından
- izin verilmeyen - başlatma yöntemlerinin çağrıları

TrendType

Trend tipini ayarlar

```
void  TrendType(
   M_TYPE_TREND    value          // yeni değer
   )
```

Parametreler

value

[in] Trend tipi için yeni değer.

Dönüş değeri

Yok.

UsedSeries

Kullanılan zaman-serilerinin bit-maskesini alır

```
int  UsedSeries()
```

Dönüş değeri

Kullanılan zaman serilerinin bit-maskesi şeklinde listesi.

Not

Bit ayarlanmışsa karşılık gelen zaman-serileri kullanılır, ayarlanmamışsa kullanılmaz.

Bitlerin zaman-serisi karşılıkları:

bit 0 - Open (açılış fiyatı) serisi,
bit 1 - High (yüksek fiyat) serisi,
bit 2 - Low (düşük fiyat) serisi,
bit 3 - Close (kapanış fiyatı) serisi,
bit 4 - Spread (alış-satış farkı) serisi,
bit 5 - Time (zaman) serisi,
bit 6 - TickVolume (tik hacmi) serisi,
bit 7 - RealVolume (işlem hacmi) serisi.

EveryTick

"Every tick" (tüm tikler) bayrağını ayarlar.

```
void  EveryTick(
   bool     value          // bayrak
   )
```

Parametreler

value

[in] Yeni bayrak değeri.

Dönüş değeri

Yok.

Not

Bayrak ayarlı değilse, işleme yöntemi sadece belirtilen sembol ve zaman-diliminde yeni bir çubuk oluşmasıyla çağrılır.

Open

Open (açılış fiyatı) serisinin elemanını indis kullanarak alır.

```
double  Open(
   int    ind          // indis
   )
```

Parametreler

ind

[in] Elemanın indisi.

Dönüş değeri

Başarı durumunda Open serisinin istenilen indis üzerindeki elemanına, aksi durumda EMPTY_VALUE değerine dönüş yapar.

Not

EMPTY_VALUE değerine iki durumda dönüş yapılır:

1. Zaman-serisi kullanılmıyordur (karşılık gelen bit değeri ayarlı değildir).
2. Eleman indisi sınırların dışındadır.

High

High (yüksek fiyat) serisinin elemanını indis kullanarak alır.

```
double  High(
   int     ind          // indis
   )
```

Parametreler

ind

[in] Elemanın indisi.

Dönüş değeri

Başarı durumunda High serisinin istenilen indis üzerindeki elemanına, aksi durumda EMPTY_VALUE değerine dönüş yapar.

Not

EMPTY_VALUE değerine iki durumda dönüş yapılır:

1. Zaman-serisi kullanılmıyordur (karşılık gelen bit değeri ayarlı değildir).
2. Eleman indisi sınırların dışındadır.

Low

Low (düşük fiyat) serisinin elemanını indis kullanarak alır.

```
double  Low(
   int     ind          // indis
   )
```

Parametreler

ind

[in] Elemanın indisi.

Dönüş değeri

Başarı durumunda Low serisinin istenilen indis üzerindeki elemanına, aksi durumda EMPTY_VALUE değerine dönüş yapar.

Not

EMPTY_VALUE değerine iki durumda dönüş yapılır:

1. Zaman-serisi kullanılmıyordur (karşılık gelen bit değeri ayarlı değildir).
2. Eleman indisi sınırların dışındadır.

Close

Close (kapanış fiyatı) serisinin elemanını indis kullanarak alır.

```
double  Close(
   int    ind          // indis
   )
```

Parametreler

ind

[in] Elemanın indisi.

Dönüş değeri

Başarı durumunda Close serisinin istenilen indis üzerindeki elemanına, aksi durumda EMPTY_VALUE değerine dönüş yapar.

Not

EMPTY_VALUE değerine iki durumda dönüş yapılır:

1. Zaman-serisi kullanılmıyordur (karşılık gelen bit değeri ayarlı değildir).
2. Eleman indisi sınırların dışındadır.

Spread

Spread (alış-satış farkı) serisinin elemanını indis kullanarak alır.

```
double  Spread(
   int    ind          // indis
   )
```

Parametreler

ind

[in] Elemanın indisi.

Dönüş değeri

Başarı durumunda Spread serisinin istenilen indis üzerindeki elemanına, aksi durumda EMPTY_VALUE değerine dönüş yapar.

Not

EMPTY_VALUE değerine iki durumda dönüş yapılır:

1. Zaman-serisi kullanılmıyordur (karşılık gelen bit değeri ayarlı değildir).
2. Eleman indisi sınırların dışındadır.

Time

Time (zaman) serisinin elemanını indis kullanarak alır.

```
datetime  Time(
   int    ind          // indis
   )
```

Parametreler

ind

[in] Elemanın indisi.

Dönüş değeri

Başarı durumunda Time serisinin istenilen indis üzerindeki elemanına, aksi durumda EMPTY_VALUE değerine dönüş yapar.

Not

EMPTY_VALUE değerine iki durumda dönüş yapılır:

1. Zaman-serisi kullanılmıyordur (karşılık gelen bit değeri ayarlı değildir).
2. Eleman indisi sınırların dışındadır.

TickVolume

TickVolume (tik hacmi) serisinin elemanını indis kullanarak alır

```
long  TickVolume(
   int    ind         // indis
   )
```

Parametreler

ind

[in] Elemanın indisi.

Dönüş değeri

Başarı durumunda TickVolume serisinin istenilen indis üzerindeki elemanına, aksi durumda EMPTY_VALUE değerine dönüş yapar.

Not

EMPTY_VALUE değerine iki durumda dönüş yapılır:

1. Zaman-serisi kullanılmıyordur (karşılık gelen bit değeri ayarlı değildir).
2. Eleman indisi sınırların dışındadır.

RealVolume

RealVolume (işlem hacmi) serisinin elemanını indis kullanarak alır.

```
long  RealVolume(
   int    ind         // indis
   )
```

Parametreler

ind

[in] Elemanın indisi.

Dönüş değeri

Başarı durumunda RealVolume serisinin istenilen indis üzerindeki elemanına, aksi durumda EMPTY_VALUE değerine dönüş yapar.

Not

EMPTY_VALUE değerine iki durumda dönüş yapılır:

1. Zaman-serisi kullanılmıyordur (karşılık gelen bit değeri ayarlı değildir).
2. Eleman indisi sınırların dışındadır.

Init

Nesneyi başlatır.

```
bool  Init(
   CSymbolInfo       symbol,    // sembol
   ENUM_TIMEFRAMES   period,    // zaman-dilimi
   double            point      // nokta
   )
```

Parametreler

symbol

[in] Sembol bilgisine erişim için CSymbolInfo nesnesinin işaretçisi.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerlerinden biri).

point

[in] 2/4-basamaklı noktanın "ağırlık" değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Symbol

Sembolü ayarlar.

```
bool  Symbol(
   string    name     // sembol
   )
```

Parametreler

name

[in] Sembol.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Nesne, başlatma sırasında tanımlanan zaman-diliminden farklı bir zaman-dilimi kullanıyorsa, çalışılan sembolün ayarlanması gereklidir.

Period

Zaman-dilimini ayarlar.

```
bool  Period(
   ENUM_TIMEFRAMES  value     // zaman-dilimi
   )
```

Parametreler

value

[in] Zaman-dilimi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Nesne, başlatma sırasında tanımlanan zaman-diliminden farklı bir zaman-dilimi kullanıyorsa, çalışılan zaman-diliminin ayarlanması gereklidir.

Magic

Uzman Danışmanın tanımlayıcısını ayarlar.

```
void  Magic(
   ulong  value     // sihirli sayı
   )
```

Parametreler

value

[in] Uzman Danışmanın tanımlayıcısı.

Dönüş değeri

Yok.

ValidationSettings

Ayarları denetler.

```
virtual bool  ValidationSettings()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

SetPriceSeries

Fiyatların dışsal zaman-serileri için işaretçiler ayarlar.

```
virtual bool  SetPriceSeries(
   CiOpen*    open,         // işaretçi
   CiHigh*    high,         // işaretçi
   CiLow*     low,          // işaretçi
   CiClose*   close         // işaretçi
   )
```

Parametreler

open

[in] Open serisinin işaretçisi.

high

[in] High serisinin işaretçisi.

low

[in] Low serisinin işaretçisi.

close

[in] Close serisinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Nesne, başlatma sırasında tanımlanan sembol veya zaman diliminden farklı sembol veya zaman dilimi kullanıyorsa, fiyatların dışsal zaman-serilerinin ayarlanması gerekir.

SetOtherSeries

Fiyatlar haricindeki dışsal zaman-serileri için işaretçiler ayarlar.

```
virtual bool  SetOtherSeries(
   CiSpread*       spread,        // işaretçi
   CiTime*         time,          // işaretçi
   CiTickVolume*   tick_volume,   // işaretçi
   CiRealVolume*   real_volume    // işaretçi
   )
```

Parametreler

spread

[in] Spread serisinin işaretçisi.

time

[in] Time serisinin işaretçisi.

tick_volume

[in] TickVolume serisinin işaretçisi.

real_volume

[in] RealVolume serisinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Nesne, başlatma sırasında tanımlanan sembol veya zaman diliminden farklı sembol veya zaman dilimi kullanıyorsa, fiyatlar haricindeki dışsal zaman-serilerinin ayarlanması gerekir.

InitIndicators

Tüm göstergeleri ve zaman-serilerini başlatır.

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // işaretçi
   )
```

Parametreler

indicators

[in] Göstergelerin ve zaman-serilerinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Nesne, başlatma sırasında tanımlanan sembol veya zaman diliminden farklı sembol veya zaman dilimi kullanıyorsa, zaman-serileri başlatılır.

InitOpen

Open (açılış fiyatı) serisini başlatır.

```
bool  InitOpen(
   CIndicators*  indicators    // işaretçi
   )
```

Parametreler

indicators

[in] Göstergelerin ve zaman-serilerinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Uzman Danışman başlatma sırasında tanımlanan sembol veya zaman diliminden farklı sembol veya zaman dilimi kullanıyorsa Open serisi başlatılır.

InitHigh

High (yüksek fiyat) serisini başlatır.

```
bool  InitHigh(
   CIndicators*  indicators    // işaretçi
   )
```

Parametreler

indicators

[in] Göstergelerin ve zaman-serilerinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Uzman Danışman başlatma sırasında tanımlanan sembol veya zaman diliminden farklı sembol veya zaman dilimi kullanıyorsa High serisi başlatılır.

InitLow

Low (düşük fiyat) serisini başlatır.

```
bool  InitLow(
   CIndicators*  indicators    // işaretçi
   )
```

Parametreler

indicators

[in] Göstergelerin ve zaman-serilerinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Uzman Danışman başlatma sırasında tanımlanan sembol veya zaman diliminden farklı sembol veya zaman dilimi kullanıyorsa Low serisi başlatılır.

InitClose

Close (kapanış fiyatı) serisini başlatır.

```
bool  InitClose(
   CIndicators*  indicators    // işaretçi
   )
```

Parametreler

indicators

[in] Göstergelerin ve zaman-serilerinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Uzman Danışman başlatma sırasında tanımlanan sembol veya zaman diliminden farklı sembol veya zaman dilimi kullanıyorsa Close serisi başlatılır.

InitSpread

Spread (alış satış farkı) serisini başlatır.

```
bool  InitSpread(
   CIndicators*  indicators    // işaretçi
   )
```

Parametreler

indicators

[in] Göstergelerin ve zaman-serilerinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Uzman Danışman başlatma sırasında tanımlanan sembol veya zaman diliminden farklı sembol veya zaman dilimi kullanıyorsa Spread serisi başlatılır.

InitTime

Time (zaman) serisini başlatır.

```
bool  InitTime(
   CIndicators*  indicators    // işaretçi
   )
```

Parametreler

indicators

[in] Göstergelerin ve zaman-serilerinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Uzman Danışman başlatma sırasında tanımlanan sembol veya zaman diliminden farklı sembol veya zaman dilimi kullanıyorsa Time serisi başlatılır.

InitTickVolume

TickVolume (tik hacmi) serisini başlatır.

```
bool  InitTickVolume(
   CIndicators*  indicators    // işaretçi
   )
```

Parametreler

indicators

[in] Göstergelerin ve zaman-serilerinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Uzman Danışman başlatma sırasında tanımlanan sembol veya zaman diliminden farklı sembol veya zaman dilimi kullanıyorsa TickVolume serisi başlatılır.

InitRealVolume

RealVolume (işlem hacmi) serisini başlatır.

```
bool  InitRealVolume(
   CIndicators*  indicators    // işaretçi
   )
```

Parametreler

indicators

[in] Göstergelerin ve zaman-serilerinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Uzman Danışman başlatma sırasında tanımlanan sembol veya zaman diliminden farklı sembol veya zaman dilimi kullanıyorsa RealVolume serisi başlatılır.

PriceLevelUnit

Fiyat seviye birimini alır

```
virtual double  PriceLevelUnit()
```

Dönüş değeri

Fiyat seviye biriminin değeri.

Not

Temel sınıfın yöntemi 2/4 basamaklı noktanın "ağırlığına" dönüş yapar.

StartIndex

Analiz edilecek başlangıç çubuğunun indisini alır.

```
virtual int  StartIndex()
```

Dönüş değeri

Analiz edilecek başlangıç çubuğunun indisi.

Not

Mevcut çubuğu analiz etme bayrağı 'true' olarak ayarlanmışsa (analize mevcut çubuktan başla), yöntem 0 dönüşü yapar. Bayrak aarlı değilse 1 dönüşü yapar (analize tamamlanan son çubuktan başla).

CompareMagic

Uzman danışmanın tanıtıcı değerini belirtilen değerle karşılaştırır.

```
virtual bool  CompareMagic(
   ulong  magic    // karşılaştırılacak değer
   )
```

Parametreler

magic

[in] Karşılaştırılacak değer.

Dönüş değeri

Değerler eşit ise 'true', aksi durumda 'false'.

CExpert

CExpert, alım-satım stratejileri için bir temel sınıftır.

Bünyesinde bazı orta seviye alım-satım yetenekleri barndırır. Bu sınıf, zaman serileri ve göstergeler ile çalışmak için kullanıma hazır algoritmalar ve alım-satım stratejileri için bir takım sanal yöntemler içerir.

Nasıl kullanılır:

1. Bir strateji algoritması hazırlayın;
2. CExpert sınıfının soyundan gelen bir sınıf örneği oluşturun;
3. Sınıfınızdaki sanal yöntemleri kendi algoritmanızı kullanarak yeniden tanımlayın.

Açıklama

CExpert sınıfı, alım-satım stratejilerinin uygulanması için bir takım sanal yöntemlerden oluşur.

Not

Bir pozisyonun hangi Uzman Danışmana tarafından yönetildiği m_symbol ve m_magic özellikleri ile belirlenir. Hedge'li modda bir sembol üzerinde birden çok pozisyon açılabildiği için m_magic değeri önemlidir.

Bildirim

```
class CExpert : public CExpertBase
```

Başlık

```
#include <Expert\Expert.mqh>
```

Kalıtım hiyerarşisi

CObject
  CExpertBase
    CExpert

Gruplarına göre Sınıf Yöntemleri

Başlatma	
Init	Sınıf örneği başlatma yöntemi
virtual InitSignal	Alım-satım Sinyali nesnesini başlatır
virtual InitTrailing	Trailing Stop (iz-süren stop) nesnesini başlatır
virtual InitMoney	Para yönetimi nesnesini başlatır
virtual InitTrade	Alım-satım nesnesini başlatır
virtual ValidationSettings	Ayarları denetler

Başlatma	
virtual InitIndicators	Göstergeleri ve zaman serilerini başlatır
virtual InitParameters	Uzman Danışman parametreleri için başlatma yöntemi
virtual Deinit	Sınıf örneği sonlandırma yöntemi
virtual DeinitSignal	Alım-satım sinyali nesnesini sonlandırır
virtual DeinitTrailing	Trailing Stop (iz-süren stop) nesnesini sonlandırır
virtual DeinitMoney	Para yönetimi nesnesini sonlandırır
virtual DeinitTrade	Alım-satım nesnesini sonlandırır
virtual DeinitIndicators	Göstergeleri ve zaman serilerini sonlandırır
Parametreler	
Magic	Uzman Danışmanın tanımlayıcısını ayarlar
MaxOrders	İzin verilen maksimum emir sayısını alır/ayarlar
OnTickProcess	"OnTick" olayını işlemek için bir bayrak ayarlar
OnTradeProcess	"OnTrade" olayını işlemek için bir bayrak ayarlar
OnTimerProcess	"OnTimer" olayını işlemek için bir bayrak ayarlar
OnChartEventProcess	"OnChartEvent" olayını işlemek için bir bayrak ayarlar
OnBookEventProcess	"OnBookEvent" olayını işlemek için bir bayrak ayarlar
Olay işleme yöntemleri	
OnTick	OnTick olay işleyicisi
OnTrade	OnTrade olay işleyicisi
OnTimer	OnTimer olay işleyicisi
OnChartEvent	OnChartEvent olay işleyicisi
OnBookEvent	OnBookEvent olay işleyicisi
Güncelleme Yöntemleri	
Refresh	Tüm verileri günceller
Processing	
Processing	Temel işleme algoritması
Piyasa Giriş Yöntemleri	
CheckOpen	Pozisyon açma koşullarını denetler
CheckOpenLong	Uzun pozisyon açma koşullarını denetler
CheckOpenShort	Kısa pozisyon açma koşullarını denetler

Başlatma	
OpenLong	Uzun pozisyon açar
OpenShort	Kısa pozisyon açar
Piyasadan Çıkış Yöntemleri	
CheckClose	Mevcut pozisyonu kapama koşullarını denetler
CheckCloseLong	Uzun pozisyon kapama koşullarını denetler
CheckCloseShort	Kısa pozisyon kapama koşullarını denetler
CloseAll	Açık pozisyonu kapar ve tüm emirleri iptal eder
Close	Açık pozisyonu kapatır
CloseLong	Uzun pozisyonu kapatır
CloseShort	Kısa pozisyonu kapatır
Pozisyon Çevirme Yöntemleri	
CheckReverse	Açık pozisyon için ters çevirme koşullarını denetler
CheckReverseLong	Uzun pozisyon için ters çevirme koşullarını denetler
CheckReverseShort	Kıza pozisyon için ters çevirme koşullarını denetler
ReverseLong	Uzun pozisyonu ters çevirir
ReverseShort	Kısa pozisyonu ters çevirir
Posisyon/Emir Taşıma Yöntemleri	
CheckTrailingStop	Pozisyon parametrelerinin değişimi için koşulları denetler
CheckTrailingStopLong	Uzun pozisyon için Trailing Stop (iz-süren stop) koşularını denetler
CheckTrailingStopShort	Kısa pozisyon için Trailing Stop koşularını denetler
TrailingStopLong	Uzun pozisyon için Trailing Stop işlemi gerçekleştirir
TrailingStopShort	Kısa pozisyon için Trailing Stop işlemi gerçekleştirir
CheckTrailingOrderLong	Limit/Stop Alış emirleri için Trailing Stop koşullarını denetler
CheckTrailingOrderShort	Limit/Stop Satış emirleri için Trailing Stop koşullarını denetler
TrailingOrderLong	Limit/Stop Alış emirleri için Trailing Stop işlemi gerçekleştirir
TrailingOrderShort	Limit/Stop Satış emirleri için Trailing Stop işlemi gerçekleştirir
Emir Silme Yöntemleri	
CheckDeleteOrderLong	Alış emrinin silinmesi için koşulları denetler

Başlatma	
CheckDeleteOrderShort	Satış emrinin silinmesi için koşulları denetler
DeleteOrders	Tüm emirleri siler
DeleteOrder	Bekleyen emri (Limit/Stop) siler
DeleteOrderLong	Bekleyen Limit/Stop Alış emrini siler
DeleteOrderShort	Bekleyen Limit/Stop Satış emrini siler
İşlem Hacmi Yöntemleri	
LotOpenLong	Uzun (alış) işlemin hacim değerini alır
LotOpenShort	Kısa (satış) işlemin hacim değerini alır
LotReverse	Pozisyonun ters çevrilmesinde kullanılan işlem hacmi değerini alır
İşlem Geçmişi Yöntemleri	
PrepareHistoryDate	Geçmiş takibi için başlangıç tarihi oluşturur
HistoryPoint	İşlem geçmişi için bir kontrol noktası oluşturur (pozisyonların, emirlerin, ve geçmiş işlemlerin sayısını kaydeder)
CheckTradeState	Mevcut durumu kaydedilmiş olanla karşılaştırır ve ilgili olay işleyicisini çağırır
Olay Bayrakları	
WaitEvent	Alım-satım olayı bekleme bayrağını ayarlar
NoWaitEvent	Alım-satım olayı bekleme bayrağını sıfırlar
Alım-Satım Olayı işleme Yöntemleri	
TradeEventPositionStopTake	"Stop Loss/Take Profit tetiklendi" olayının işleyicisi
TradeEventOrderTriggered	"Bekleyen Emir Tetiklendi" olayının işleyicisi
TradeEventPositionOpened	"Pozisyon Açıldı" olayının işleyicisi
TradeEventPositionVolumeChanged	"Pozisyon Hacmi Değişti" olayının işleyicisi
TradeEventPositionModified	"Pozisyon Değiştirildi" olayının işleyicisi
TradeEventPositionClosed	"Pozisyon Kapandı" olayının işleyicisi
TradeEventOrderPlaced	"Bekleyen Emir Oluşturuldu" olayının işleyicisi
TradeEventOrderModified	"Bekleyen Emir Değiştirildi" olayının işleyicisi
TradeEventOrderDeleted	"Bekleyen Emir Silindi" olayının işleyicisi
TradeEventNotIdentified	Tanımlanmayan olay için olay işleyici

Başlatma	
Hizmet yöntemleri	
TimeframeAdd	İzlenecek zaman-dilimi ekler
TimeframesFlags	Zaman-dilimi bayraklarını ayarlar
SelectPosition	Çalışılacak bir pozisyon seçer

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

Init

Sınıf örneği başlatma yöntemi.

```
bool  Init(
   string              symbol,         // sembol
   ENUM_TIMEFRAMES     period,         // zaman-dilimi
   bool                every_tick,     // bayrak
   ulong               magic           // uzmanın tanıtıcı değeri
   )
```

Parametreler

symbol

[in] Sembol.

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımı).

every_tick

[in] Bayrak.

magic

[in] Uzman Danışmanın tanıtıcı değeri (Magic number).

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

every_tick parametresi true olarak ayarlanmışsa Processing() çalışılan sembolün yöntemi her bir tik olayı için çağrılır. Aksi durumda, Processing() yöntemi sadece yeni çubuk oluştuğunda çağrılacaktır.

Magic

Uzman Danışmanın tanımlayıcısını ayarlar.

```
void  Magic(
   ulong  value     // yeni değer
   )
```

Parametreler

value

[in] Uzman Danışmanın tanımlayıcısının yeni değeri.

Dönüş değeri

Yok.

Not

Uzman danışmanın tanımlayıcısını değiştirdiğinizde tüm yardımcı nesneler (sınıflar) de aynı tanımlayıcıyı kullanır.

Uygulama

```
//+------------------------------------------------------------------+
//| Nesne ve onun tüm bağımlı nesneleri için tanıtıcı değer ayarla   |
//| INPUT:  value - tanıtıcının yeni değeri.                         |
//| OUTPUT: yok.                                                     |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
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

Alım-satım Sinyali nesnesini başlatır.

```
virtual bool  InitSignal(
   CExpertSignal*     signal=NULL,        // işaretçi
   )
```

Parametreler

signal

[in] CExpertSignal sınıf örneğinin veya onu türevlerinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

signal parametresinin değeri NULL ise, CExpertSignal sınıfı kullanılır ve hiçbir eylem gerçekleştirmez.

InitTrailing

Trailing Stop (iz-süren stop) nesnesini başlatır.

```
virtual bool  InitTrailing(
   CExpertTrailing*     trailing=NULL,        // işaretçi
   )
```

Parametreler

trailing

[in] CExpertTrailing sınıf örneğinin veya onu türevlerinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

'trailing' parametresinin değeri NULL ise, ExpertTrailing sınıfı kullanılır ve hiçbir eylem gerçekleştirmez.

InitMoney

Para yönetimi nesnesini başlatır.

```
virtual bool  InitMoney(
   CExpertMoney*     money=NULL,         // işaretçi
   )
```

Parametreler

money

[in] CExpertMoney sınıf örneğinin veya onu türevlerinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

'money' parametresinin değeri NULL ise, CExpertMoney çağrıları minimum lot ile kullanılır.

InitTrade

Alım-satım nesnesini başlatır.

```
virtual bool  InitTrade(
   ulong            magic,       // uzman tanıtıcı değeri
   CExpertTrade*    trade=NULL   // işaretçi
   )
```

Parametreler

magic

[in] Uzman danışmanın tanıtıcı değeri (alım satı isteklerinde kullanılır).

trade

[in] CExpertTrade nesnesinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Deinit

Sınıf örneği sonlandırma yöntemi.

```
virtual void Deinit()
```

Dönüş değeri

Yok.

OnTickProcess

OnTick olayını işlemek için bir bayrak ayarlar.

```
void  OnTickOProcess(
   bool     value        // bayrak
   )
```

Parametreler

value

[in] OnTick olayını işlemek için bayrak.

Dönüş değeri

Yok.

Not

Bayrak değeri ('value' parametresi) 'true' ise OnTick olayı işlenir, varsayılan olarak, bayrak 'true' değeriyle ayarlıdır.

OnTradeProcess

OnTrade olayını işlemek için bir bayrak ayarlar.

```
void  OnTradeProcess(
   bool     value        // bayrak
   )
```

Parametreler

value

[in] OnTrade olayını işlemek için bayrak.

Dönüş değeri

Yok.

Not

Bayrak değeri ('value' parametresi) 'true' ise OnTrade olayı işlenir, varsayılan olarak, bayrak 'false' değeriyle ayarlıdır.

OnTimerProcess

OnTimer olayını işlemek için bir bayrak ayarlar.

```
void  OnTimerProcess(
   bool     value        // bayrak
   )
```

Parametreler

value

[in] OnTimer olayının işlenmesi için gerekli bayrak.

Dönüş değeri

Yok.

Not

Bayrak değeri ('value' parametresi) 'true' ise OnTimer olayı işlenir, varsayılan olarak, bayrak 'false' değeriyle ayarlıdır.

OnChartEventProcess

OnChartEvent olayını işlemek için bir bayrak ayarlar.

```
void  OnChartEventProcess(
   bool     value        // bayrak
   )
```

Parametreler

value

[in] OnChartEvent olayını işlemek için bayrak.

Dönüş değeri

Yok.

Not

Bayrak değeri ('value' parametresi) 'true' ise OnChartEvent olayı işlenir, varsayılan olarak, bayrak 'false' değeriyle ayarlıdır.

OnBookEventProcess

OnBookEvent olayını işlemek için bir bayrak ayarlar.

```
void  OnChartEventProcess(
   bool     value        // bayrak
   )
```

Parametreler

value

[in] OnBookEvent olayının işlenmesi için gerekli bayrak.

Dönüş değeri

Yok.

Not

Bayrak değeri ('value' parametresi) 'true' ise OnBookEvent olayı işlenir, varsayılan olarak, bayrak 'false' değeriyle ayarlıdır.

MaxOrders (Get Yöntemi)

İzin verilen maksimum emir sayısını alır.

```
int  MaxOrders()
```

Dönüş değeri

İzin verilen maksimum emir sayısı.

MaxOrders (Set Yöntemi)

İzin verilen maksimum emir sayısını ayarlar.

```
void  MaxOrders(
   int      max_orders         // yeni değer
   )
```

Parametreler

max_orders

[in] İzin verilen maksimum emir sayısı için yeni değer.

Dönüş değeri

Yok.

Not

Varsayılan olarak izin verilen maksimum emir sayısı = 1.

Signal

Alım-satım sinyali nesnesinin işaretçisini alır.

```
CExpertSignal*  Signal() const
```

Dönüş değeri

Alım-satım sinyali nesnesinin işaretçisi.

ValidationSettings

Ayarları denetler.

```
virtual bool  ValidationSettings()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Ayrıca, tüm Uzman Danışman nesnelerinin de ayarlarını denetler.

InitIndicators

Tüm göstergeleri ve zaman-serilerini başlatır.

```
virtual bool  InitIndicators(
   CIndicators*  indicators=NULL    // işaretçi
   )
```

Parametreler

indicators

[in] Göstergelerin ve zaman-serilerinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Nesne, başlatma sırasında tanımlanan sembol veya zaman diliminden farklı sembol veya zaman dilimi kullanıyorsa, zaman-serileri başlatılır. Devamlı olarak, 'alım-satım sinyali', 'trailing stop' ve 'para yönetimi' nesnelerinin sanal yöntemlerini (InitIndicators()) çağırır.

OnTick

OnTick olayının sanal olay işleyicisi.

```
virtual void  OnTick()
```

Dönüş değeri

Yok.

OnTrade

OnTrade olayının sanal olay işleyicisi.

```
virtual void  OnTrade()
```

Dönüş değeri

Yok.

OnTimer

OnTimer olayının sanal olay işleyicisi.

```
virtual void  OnTimer()
```

Dönüş değeri

Yok.

OnChartEvent

OnChartEvent olayının sanal olay işleyicisi.

```
virtual void  OnChartEvent(
   const int     id,       // olay tanımlayıcı
   const long&   lparam,   // long parametre
   const double  dparam,   // double parametre
   const string  sparam    // string parametre
   )
```

Parametreler

id

[in] Olay tanımlayıcısı.

lparam

[in] long tipli olay parametresi.

dparam

[in] double tipli olay parametresi.

sparam

[in] string tipli olay parametresi.

Dönüş değeri

Yok.

OnBookEvent

OnBookEvent olayının sanal olay işleyicisi.

```
virtual void  OnBookEvent(
   const string&     symbol       // sembol
)
```

Parametreler

symbol

[in] OnBookEvent olayı için sembol.

Dönüş değeri

Yok.

InitParameters

Uzman Danışman parametrelerini başlatır.

```
virtual bool  InitParameters()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

CExpert sınıfının InitParameters() yöntemi eylem gerçekleştirmez ve her zaman 'true' dönüşü yapar.

DeinitTrade

Alım-satım nesnesini sonlandırır.

```
virtual void  DeinitTrade()
```

Dönüş değeri

Yok.

DeinitSignal

Sinyal nesnesini sonlandırır.

```
virtual void  DeinitSignal()
```

Dönüş değeri

Yok.

DeinitTrailing

Trailing (iz-süren) nesnesini sonlandırır.

```
virtual void  DeinitTrailing()
```

Dönüş değeri

Yok.

DeinitMoney

Para yönetimi nesnesini sonlandırır.

```
virtual void  DeinitMoney()
```

Dönüş değeri

Yok.

DeinitIndicators

Tüm göstergeleri ve zaman serilerini sonlandırır

```
virtual void  DeinitIndicators()
```

Dönüş değeri

Yok.

Not

Ayrıca tüm yardımcı nesnelerin de gösterge ve zaman serilerini kapatır.

Refresh

Tüm verileri günceller.

```
virtual bool  Refresh()
```

Dönüş değeri

Daha ileri bir tik işleme gerekiyorsa 'true', aksi durumda 'false'.

Not

Tik işleme ihtiyacının belirlenmesini sağlar. İhtiyaç duyulduğunda tüm fiyatları, zaman-serilerini ve gösterge verilerini günceller ve 'true' dönüşü yapar.

Uygulama

```
//+------------------------------------------------------------------+
//| İşlenmesi için veriyi günceller                                  |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem başarılıysa -true, değilse -false.                 |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::Refresh()
  {
   MqlDateTime time;
//--- yenileme oranı
   if(!m_symbol.RefreshRates()) return(false);
//--- işleme ihtiyacını denetle
   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;
//--- göstergeleri yenile
   m_indicators.Refresh();
//--- tamam
   return(true);
  }
```

Processing

Temel işleme algoritması.

```
virtual bool  Processing()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Şu adımları uygular:

1. Sembol üzerinde açık pozisyonların mevcudiyetini kontrol et. Açık bir pozisyon yoksa 2, 3 ve 4 numaralı adımları atla.
2. Açık pozisyon için ters çevirme koşullarını denetle (CheckReverse() yöntemi). Pozisyon "ters çevrilmişse" çık.
3. Pozisyon kapama koşullarını kontrol eder (CheckClose() yöntemi). Pozisyon kapalıysa 4 numaralı adım atla.
4. Pozisyon parametrelerinin değişimi için koşulları denetler (CheckTrailingStop() yöntemi). Pozisyon parametreleri değiştirilmişse çık.
5. Bekleyen emirlerin varlığını araştır. Bekleyen emir yoksa 9 numaralı adıma geç.
6. Emir silme için koşulları denetle (bekleyen alış emirleri için CheckDeleteOrderLong() veya bekleyen satış emirleri için CheckDeleteOrderShort()). Emir silinmişse 9 numaralı adıma geçer.
7. Bekleyen emirlerin parametrelerini değiştirmek için koşulları denetle (alış emirleri için CheckTrailingOrderLong(), satış emirleri için CheckTrailingOrderShort()). Parametreler değiştirilmişse çık.
8. Çık.
9. Pozisyon açma koşullarını denetle (CheckOpen()) yöntemi).

Kendi algoritmanızı yazmak istediğinizde, mirasçı sınıfın Processing() yöntemini geçersiz kılmanız gerekir.

Uygulama

```
//+------------------------------------------------------------------+
//| Ana fonksiyon                                                    |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::Processing()
  {
//--- açık pozisyonları denetle
   if(m_position.Select(m_symbol.Name()))
     {
      //--- açık pozisyon mevcut
      //--- pozisyonu ters çevirmek için koşulları denetle
      if(CheckReverse()) return(true);
      //--- pozisyon kapama / bekleyen emirleri silme koşullarını denetle
      if(!CheckClose())
        {
         //--- pozisyon değişim için koşullarını denetle
         if(CheckTrailingStop()) return(true);
         //--- işlemsiz dönüş yap
         return(false);
        }
     }
//--- bekleyen emir yerleştirilmiş mi denetle
   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
           {
            //--- bekleyen uzun (alış) emrin silinmesi için koşulları denetle
            if(CheckDeleteOrderLong()) return(true);
            //--- bekleyen uzun (alış) emrin değiştirilmesi için koşulları denetle
            if(CheckTrailingOrderLong()) return(true);
           }
         else
           {
            //--- bekleyen kısa (satış) emrin silinmesi için koşulları denetle
            if(CheckDeleteOrderShort()) return(true);
            //--- bekleyen kısa (satış) emrin değiştirilmesi için koşulları denetle
            if(CheckTrailingOrderShort()) return(true);
           }
         //--- işlemsiz dönüş yap
         return(false);
        }
     }
//--- pozisyon açma / bekleyen emir değiştirme koşullarını denetle
   if(CheckOpen()) return(true);
//--- işlemsiz dönüş yap
   return(false);
  }
```

SelectPosition

Selects a position to work with.

```
void  SelectPosition()
```

Return Value

No.

Implementation

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

Pozisyon açma koşullarını denetler.

```
virtual bool  CheckOpen()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Uzun (CheckOpenLong()) ve kısa (CheckOpenShort()) pozisyon açma koşullarını denetler.

Uygulama

```
//+------------------------------------------------------------------+
//| Pozisyon açma veya limit/stop emri ayarlama için denetle         |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CheckOpen()
  {
   if(CheckOpenLong())  return(true);
   if(CheckOpenShort()) return(true);
//--- işlemsiz dönüş yap
   return(false);
  }
```

CheckOpenLong

Uzun pozisyon açma koşullarını denetler.

```
virtual bool  CheckOpenLong()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Uzun pozisyonu açmak için gerekli koşulları denetler (sinyal nesnesinin CheckOpenLong() yöntemi) ve koşullar sağlanıyorsa pozisyon açar (OpenLong() yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Uzun pozisyon açma veya limit/stop emirleri ayarlama için denetle|
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CheckOpenLong()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- uzun pozisyom açma koşullarını denetle
   if(m_signal.CheckOpenLong(price,sl,tp,expiration))
     {
      if(!m_trade.SetOrderExpiration(expiration))
        {
         m_expiration=expiration;
        }
      return(OpenLong(price,sl,tp));
     }
//--- işlemsiz dönüş yap
   return(false);
  }
```

CheckOpenShort

Kısa pozisyon açma koşullarını denetler.

```
virtual bool  CheckOpenShort()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Kısa pozisyonu açmak için gerekli koşulları denetler (sinyal nesnesinin CheckOpenShort() yöntemi) ve koşullar sağlanıyorsa pozisyon açar (OpenShort() yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Kısa pozisyon açmak veya limit/stop emiri ayarlamak için denetle |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CheckOpenShort()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- kısa pozisyom açma koşullarını denetle
   if(m_signal.CheckOpenShort(price,sl,tp,expiration))
     {
      if(!m_trade.SetOrderExpiration(expiration))
        {
         m_expiration=expiration;
        }
      return(OpenShort(price,sl,tp));
     }
//--- işlemsiz dönüş yap
   return(false);
  }
```

OpenLong

Bir uzun pozisyon açar

```
virtual bool  OpenLong(
   double    price,    // fiyat
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Parametreler

price

[in] Fiyat.

sl

[in] Stop Loss (zararı durdur) fiyatı.

tp

[in] Take Profit (kar al) fiyatı.

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

İşlem hacmi değerini alır (LotOpenLong(...) yöntemi) ve hacim sıfırdan büyükse bir uzun pozisyon açar (Trade nesnesinin Buy() yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Uzun pozisyon açılışı veya limit/stop emrinin değişimi           |
//| INPUT:  price - fiyat,                                           |
//|         sl    - stop loss,                                       |
//|         tp    - take profit.                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::OpenLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- açılış için hacim al
   double lot=LotOpenLong(price,sl);
//--- açılış hacmini denetle
   if(lot==0.0) return(false);
//---
   return(m_trade.Buy(lot,price,sl,tp));
  }
```

OpenShort

Bir kısa pozisyon açar.

```
virtual bool  OpenShort(
   double    price,    // fiyat
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Parametreler

price

[in] Fiyat.

sl

[in] Stop Loss (zararı durdur) fiyatı.

tp

[in] Take Profit (kar al) fiyatı.

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

İşlem hacmi değerini alır (LotOpenShort(...) yöntemi) ve hacim sıfırdan büyükse bir kısa pozisyon açar (Trade nesnesinin Sell() yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Kısa pozisyon açılışı veya limit/stop emrinin ayarlanması        |
//| INPUT:  price - fiyat,                                           |
//|         sl    - stop loss,                                       |
//|         tp    - take profit.                                     |
//| OUTPUT: işlem başarılıysa - true, değilse - false.               |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::OpenShort(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- açılış için hacim al
   double lot=LotOpenShort(price,sl);
//--- açılış hacmini denetle
   if(lot==0.0) return(false);
//---
   return(m_trade.Sell(lot,price,sl,tp));
  }
```

CheckReverse

Açık pozisyon için ters çevirme koşullarını denetler.

```
virtual bool  CheckReverse()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Uzun (CheckReverseLong()) ve kısa (CheckReverseShort()) pozisyon çevirme koşullarını denetler.

Uygulama

```
//+------------------------------------------------------------------+
//| Pozisyon ters çevirme için koşulları denetle                     |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CheckReverse()
  {
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- uzun pozisyonu ters çevirmek için koşulları denetle
      if(CheckReverseLong())  return(true);
     }
   else
      //--- kısa pozisyonu ters çevirmek için koşulları denetle
      if(CheckReverseShort()) return(true);
//--- işlemsiz dönüş yap
   return(false);
  }
```

CheckReverseLong

Uzun pozisyon için ters çevirme koşullarını denetler.

```
virtual bool  CheckReverseLong()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Uzun pozisyonu ters çevirmek için gerekli koşulları denetler (sinyal nesnesinin CheckReverseLong() yöntemi) ve koşullar sağlanıyorsa mevcut uzun pozisyonu ters çevirir (ReverseLong() yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Uzun pozisyonu ters çevirme için denetle                         |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CheckReverseLong()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- uzun pozisyon ters çevirme sinyalini denetle
   if(m_signal.CheckReverseLong(price,sl,tp,expiration)) return(ReverseLong(price,sl,t
//--- işlemsiz dönüş yap
   return(false);
  }
```

CheckReverseShort

Kıza pozisyon için ters çevirme koşullarını denetler.

```
virtual bool  CheckReverseLong()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Kısa pozisyonu ters çevirmek için gerekli koşulları denetler (sinyal nesnesinin CheckReverseShort() yöntemi) ve koşullar sağlanıyorsa mevcut kısa pozisyonu ters çevirir (ReverseShort() yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Kısa pozisyonu ters çevirme için denetle                         |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CheckReverseShort()
  {
   double   price=EMPTY_VALUE;
   double   sl=0.0;
   double   tp=0.0;
   datetime expiration=TimeCurrent();
//--- kısa pozisyon ters çevirme sinyalini denetle
   if(m_signal.CheckReverseShort(price,sl,tp,expiration)) return(ReverseShort(price,sl
//--- işlemsiz dönüş yap
   return(false);
  }
```

ReverseLong

Uzun pozisyonun ters çevirme işlemini gerçekleştirir.

```
virtual bool  ReverseLong(
   double    price,    // fiyat
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Parametreler

price

[in] Fiyat.

sl

[in] Stop Loss (zararı durdur) fiyatı.

tp

[in] Take Profit (kar al) fiyatı.

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Ters çevirme işleminin hacmini alır (LotReverse() yöntemi) ve hacim değeri sıfırdan büyükse uzun işlemi tersine çevirir (Trade nesnesinin Sell() yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Uzun pozisyonu ters çevirme                                      |
//| INPUT:  price - fiyat,                                           |
//|         sl    - stop loss,                                       |
//|         tp    - take profit.                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::ReverseLong(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- Ters işlem için lot değeri al
   double lot=LotReverse(sl);
//--- lotu denetle
   if(lot==0.0) return(false);
//---
   return(m_trade.Sell(lot,price,sl,tp));
  }
```

ReverseShort

Kısa pozisyonu ters çevirir.

```
virtual bool  ReverseShort(
   double    price,    // fiyat
   double    sl,       // Stop Loss
   double    tp        // Take Profit
   )
```

Parametreler

price

[in] Fiyat.

sl

[in] Stop Loss (zararı durdur) fiyatı.

tp

[in] Take Profit (kar al) fiyatı.

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Ters çevirme işleminin hacmini alır (LotReverse() yöntemi) ve hacim değeri sıfırdan büyükse kısa işlemi tersine çevirir (Trade nesnesinin Buy() yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| ısa pozisyonu ters çevirme                                       |
//| INPUT:  price - fiyat,                                           |
//|         sl    - stop loss,                                       |
//|         tp    - take profit.                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::ReverseShort(double price,double sl,double tp)
  {
   if(price==EMPTY_VALUE) return(false);
//--- Ters işlem için lot değeri al
   double lot=LotReverse(sl);
//--- lotu denetle
   if(lot==0.0) return(false);
//---
   return(m_trade.Buy(lot,price,sl,tp));
  }
```

CheckClose

Pozisyon kapama koşullarını denetler.

```
virtual bool  CheckClose()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

1. Uzman Danışmanın Stop Out (piyasadan çıkış) koşullarını denetler (para yönetimi nesnesinin metodu CheckClose() çağrısı yapar). Koşul sağlanıyorsa pozisyonu kapatır ve tüm emirleri siler (CloseAll(...)) ve çıkar.
2. Kısa ve uzun, her iki durum için de koşul denetimi yapar (CheckCloseLong() ve CheckCloseShort() yöntemleri) ve eğer pozisyon kapatılmışsa tüm emirleri siler (DeleteOrders() yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Pozisyon kapanması ve limit/stop emirlerinin silinmesini denetle |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CheckClose()
  {
   double lot;
//--- pozisyon, çağrıdan önce seçilmeli
   if((lot=m_money.CheckClose(GetPointer(m_position)))!=0.0)
      return(CloseAll(lot));
//--- pozisyon tipini kontrol et
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- uzun pozisyon kapama / bekleyen alış emirlerini silme koşullarını denetle
      if(CheckCloseLong())
        {
         DeleteOrders();
         return(true);
        }
     }
   else
     {
      //--- kısa pozisyon kapama / bekleyen satış emirlerini silme koşullarını denetle
      if(CheckCloseShort())
        {
         DeleteOrders();
         return(true);
        }
     }
//--- işlemsiz dönüş yap
   return(false);
  }
```

CheckCloseLong

Uzun pozisyon kapama koşullarını denetler.

```
virtual bool  CheckCloseLong()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Uzun pozisyonu kapamak için gerekli koşulları denetler (sinyal nesnesinin CheckCloseLong() yöntemi) ve koşullar sağlanıyorsa açık pozisyonu siler (CloseLong(...) yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Uzun pozisyonu kapama veya limit/stop emrini silme için denetle  |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CheckCloseLong()
  {
   double price=EMPTY_VALUE;
//--- uzun pozisyon kapama koşullarını denetle
   if(m_signal.CheckCloseLong(price))
      return(CloseLong(price));
//--- işlemsiz dönüş yap
   return(false);
  }
```

CheckCloseShort

Kısa pozisyon kapama koşullarını denetler.

```
virtual bool  CheckCloseShort()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Kısa pozisyonu kapamak için gerekli koşulları denetler (sinyal nesnesinin CheckCloseShort() yöntemi) ve koşullar sağlanıyorsa açık pozisyonu siler (CloseShort() yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Kısa pozisyon kapama veya limit/stop emri silme için denetle     |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CheckCloseShort()
  {
   double price=EMPTY_VALUE;
//---kısa pozisyon kapma için kontrol et
   if(m_signal.CheckCloseShort(price))
      return(CloseShort(price));
//--- işlemsiz dönüş yap
   return(false);
  }
```

CloseAll

Pozisyonların bir kısmını veya tamamını kapatır.

```
virtual bool  CloseAll(
   double    lot    // lot
   )
```

Parametreler

lot

[in] Pozisyonlardan azaltılacak lot sayısı.

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Pozisyon kapama işlemi için "Netleştirme" modunda CExpertTrade::Buy veya CExpertTrade::Sell yöntemlerini, "Hedge'li" modda ise CTrade::PositionClose yöntemini kullanabilirsiniz (bu yöntem netleştirme modunda da kullanılabilir). DeleteOrders() yöntemiyle emirleri silebilirsiniz.

Uygulama

```
//+------------------------------------------------------------------+
//| Pozisyonları kapa ve emirleri sil                                |
//| INPUT:  lot - kapama hacmi.                                      |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CloseAll(double lot)
  {
   bool result;
//--- pozisyon kapama işlemlerini denetle
   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

Açık pozisyonu kapatır.

```
virtual bool  Close()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Pozisyonu kapatır (CTrade sınıf nesnesinin PositionClose() yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Pozisyon kapama                                                  |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::Close()
  {
   return(m_trade.PositionClose(m_symbol.Name()));
  }
```

CloseLong

Uzun pozisyonu kapar.

```
virtual bool  CloseLong(
   double    price    // fiyat
   )
```

Parametreler

price

[in] Fiyat.

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Uzun pozisyonu kapar (CTrade sınıf nesnesinin Sell(...) yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Uzun pozisyon kapama                                             |
//| INPUT:  price - kapama fiyatı.                                   |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CloseLong(double price)
  {
   if(price==EMPTY_VALUE) return(false);
//---
   return(m_trade.Sell(m_position.Volume(),price,0,0));
  }
```

CloseShort

Kısa pozisyonu kapatır.

```
virtual bool  CloseShort(
   double    price    // fiyat
   )
```

Parametreler

price

[in] Fiyat.

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Kısa pozisyonu kapar (CTrade sınıf nesnesinin Buy(...) yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Kısa pozisyon kapama                                             |
//| INPUT:  price - kapama fiyatı.                                   |
//| OUTPUT: işlem başarılıysa - true, değilse - false.               |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CloseShort(double price)
  {
   if(price==EMPTY_VALUE) return(false);
//---
   return(m_trade.Buy(m_position.Volume(),price,0,0));
  }
```

CheckTrailingStop

Açık pozisyon için Trailing (iz-süren) Stop koşularını denetler

```
virtual bool  CheckTrailingStop()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Açık pozisyon için Trailing Stop koşularını denetler (uzun ve kısa pozisyonla için CheckTrailingStopLong() veya CheckTrailingStopShort() yöntemleri).

Uygulama

```
//+------------------------------------------------------------------+
//| Trailing stop/profit pozisyonunu denetle                         |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStop()
  {
//--- pozisyon, çağrıdan önce seçilmeli
   if(m_position.PositionType()==POSITION_TYPE_BUY)
     {
      //--- uzun pozisyonu değiştirmek için koşulları denetle
      if(CheckTrailingStopLong()) return(true);
     }
   else
     {
      //--- kısa pozisyonu değiştirmek için koşulları denetle
      if(CheckTrailingStopShort()) return(true);
     }
//--- işlemsiz dönüş yap
   return(false);
  }
```

CheckTrailingStopLong

Açık uzun pozisyon için Trailing (iz-süren) Stop koşularını denetler

```
virtual bool  CheckTrailingStopLong()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Açık uzun pozisyon için Trailing Stop koşularını denetler (Expert Trailing nesnesinin CheckTrailingStopLong(...) yöntemi). Koşullar sağlanmışsa pozisyon parametrelerini değiştirir (TrailingStopLong(...) yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Uzun iz-süren stop/profit pozisyon koşularını denetle            |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStopLong()
  {
   double sl=EMPTY_VALUE;
   double tp=EMPTY_VALUE;
//--- Uzun iz-süren limit/stop işlemleri için denetle
   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();
      //--- Uzun iz-süren limit/stop işlemleri için denetle
      return(TrailingStopLong(sl,tp));
     }
//--- işlemsiz dönüş yap
   return(false);
  }
```

CheckTrailingStopShort

Açık kısa pozisyon için Trailing (iz-süren) Stop koşularını denetler

```
virtual bool  CheckTrailingStopShort()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Açık kısa pozisyon için Trailing Stop koşularını denetler (Expert Trailing nesnesinin CheckTrailingStopShort(...) yöntemi). Koşullar sağlanmışsa pozisyon parametrelerini değiştirir (TrailingStopShort(...) yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Kısa iz-süren stop/profit pozisyon koşularını denetle            |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingStopShort()
  {
   double sl=EMPTY_VALUE;
   double tp=EMPTY_VALUE;
//--- Kısa iz-süren stop/profit işlemlerini denetle
   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();
      //--- Kısa iz-süren stop/profit işlemleri
      return(TrailingStopShort(sl,tp));
     }
//--- işlemsiz dönüş yap
   return(false);
  }
```

TrailingStopLong

Açık uzun pozisyonun parametrelerini değiştirir

```
virtual bool  TrailingStopLong(
   double    sl,    // Stop Loss
   double    tp,    // Take Profit
   )
```

Parametreler

sl

[in] Stop Loss (zararı durdur) fiyatı.

tp

[in] Take Profit (kar al) fiyatı.

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Bu fonksiyon açık durumdaki uzun pozisyonların parametrelerini değiştirir (CTrade sınıf örneğinin PositionModify(...) yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Uzun iz-süren stop/profit pozisyonu                              |
//| INPUT:  sl - yeni stop loss,                                     |
//|         tp - yeni take profit                                    |
//| OUTPUT: işlem başarılıysa - true, değilse - false.               |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::TrailingStopLong(double sl,double tp)
  {
   return(m_trade.PositionModify(m_symbol.Name(),sl,tp));
  }
```

TrailingStopShort

Açık kısa pozisyonun parametrelerini değiştirir

```
virtual bool  TrailingStopLong(
   double    sl,    // Stop Loss
   double    tp,    // Take Profit
   )
```

Parametreler

sl

[in] Stop Loss (zararı durdur) fiyatı.

tp

[in] Take Profit (kar al) fiyatı.

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Bu fonksiyon açık durumdaki kısa pozisyonun parametrelerini değiştirir (CTrade sınıf örneğinin PositionModify(...) yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Kısa iz-süren stop/profit pozisyonu                              |
//| INPUT:  sl - yeni stop loss,                                     |
//|         tp - yeni take profit                                    |
//| OUTPUT: işlem başarılıysa - true, değilse - false.               |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::TrailingStopShort(double sl,double tp)
  {
   return(m_trade.PositionModify(m_symbol.Name(),sl,tp));
  }
```

CheckTrailingOrderLong

Bekleyen uzun emirler için Trailing (iz-süren) Stop koşullarını denetler.

```
virtual bool  CheckTrailingOrderLong()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Bekleyen limit/stop alım emirleri için Trailing Stop koşullarını denetler (Alım-satım Sinyalleri nesnesinin CheckTrailingOrderLong() yöntemi) ve gerektiğinde emir parametrelerini değiştirir (TrailingOrderLong(...) yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Uzun limit/stop emirler için taşıma koşullarını denetle          |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingOrderLong()
  {
   double price;
//--- uzun pozisyon değiştirme koşullarını denetle
   if(m_signal.CheckTrailingOrderLong(GetPointer(m_order),price))
      return(TrailingOrderLong(m_order.PriceOpen()-price));
//--- işlemsiz dönüş yap
   return(false);
  }
```

CheckTrailingOrderShort

Bekleyen Limit/Stop kısa emirler için Trailing (iz-süren) Stop koşullarını denetler.

```
virtual bool  CheckTrailingOrderShort()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Bekleyen limit/stop satış emirleri için Trailing Stop koşullarını denetler (İşlem Sinyalleri nesnesinin CheckTrailingOrderShort() yöntemi) ve gerektiğinde emir parametrelerini değiştirir (TrailingOrderShort(...) yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Kısa limit/stop emirleri için taşıma koşullarını denetle         |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CheckTrailingOrderShort()
  {
   double price;
//--- kısa pozisyon kapama koşullarını denetle
   if(m_signal.CheckTrailingOrderShort(GetPointer(m_order),price))
      return(TrailingOrderShort(m_order.PriceOpen()-price));
//--- işlemsiz dönüş yap
   return(false);
  }
```

TrailingOrderLong

Bekleyen Limit/Stop alış emirlerinin parametrelerini değiştirir.

```
virtual bool  TrailingOrderLong(
   double    delta    // delta
   )
```

Parametreler

delta

[in] Fiyat değişimi.

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Bekleyen Limit/Stop alış emirlerinin parametrelerini değiştirir (CTrade sınıf örneğinin OrderModify(...) yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| İz-süren uzun limit/stop emirleri                                |
//| INPUT:  delta - fiyat değişimi.                                  |
//| OUTPUT: işlem başarılıysa - true, değilse - false.               |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
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;
//--- uzun emri değiştir
   return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpira
  }
```

TrailingOrderShort

Bekleyen Limit/Stop satış emirlerinin parametrelerini değiştirir.

```
virtual bool  TrailingOrderShort(
   double    delta    // delta
   )
```

Parametreler

delta

[in] Fiyat değişimi.

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Bekleyen Limit/Stop satış emirlerinin parametrelerini değiştirir (CTrade sınıf örneğinin OrderModify(...) yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| İz-süren kısa limit/stop emirleri                                |
//| INPUT:  delta - fiyat değişimi.                                  |
//| OUTPUT: işlem başarılıysa - true, değilse - false.               |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
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;
//--- kısa emri değiştir
   return(m_trade.OrderModify(ticket,price,sl,tp,m_order.TypeTime(),m_order.TimeExpira
  }
```

CheckDeleteOrderLong

Bekleyen uzun (alış) Limit/Stop emirlerin silinmesi için koşulları denetler.

```
virtual bool  CheckDeleteOrderLong()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

emrin zaman aşımını denetler. Uzun pozisyonu kapamak için gerekli koşulları denetler (sinyal nesnesinin CheckCloseLong() yöntemi) ve koşullar sağlanıyorsa açık pozisyonu siler (DeleteOrderLong() yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Kısa limit/stop emrini silme için denetle                        |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CheckDeleteOrderLong()
  {
   double price;
//--- uzun pozisyon kapama koşullarını denetle
   if(m_expiration!=0 && TimeCurrent()>m_expiration)
     {
      m_expiration=0;
      return(DeleteOrderLong());
     }
   if(m_signal.CheckCloseLong(price))
      return(DeleteOrderLong());
//--- işlemsiz dönüş yap
   return(false);
  }
```

CheckDeleteOrderShort

Bekleyen kısa (satış) Limit/Stop emirlerin silinmesi için koşulları denetler.

```
virtual bool  CheckDeleteOrderShort()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

emrin zaman aşımını denetler. Kısa pozisyonu kapamak için gerekli koşulları denetler (sinyal nesnesinin CheckCloseShort(...) yöntemi) ve koşullar sağlanıyorsa açık pozisyonu siler (DeleteOrderShort() yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Kısa limit/stop emrini silmek için denetle                       |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem gerçekleşmişse - true, aksi durumda - false.       |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::CheckDeleteOrderShort()
  {
   double price;
//--- kısa pozisyon kapama koşullarını denetle
   if(m_expiration!=0 && TimeCurrent()>m_expiration)
     {
      m_expiration=0;
      return(DeleteOrderShort());
     }
   if(m_signal.CheckCloseShort(price))
      return(DeleteOrderShort());
//--- işlemsiz dönüş yap
   return(false);
  }
```

DeleteOrders

Tüm emirleri siler.

```
virtual bool  DeleteOrders()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Tüm emirleri siler (tüm emirler için DeleteOrder() yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| tüm limit/stop emirlerini sil                                    |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem başarılıysa - true, değilse - false.               |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
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

Bekleyen Limit/Stop emrini siler.

```
virtual bool  DeleteOrder()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Bekleyen Limit/Stop emrini siler (CTrade sınıfının OrderDelete(...) yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| limit/stop emrini sil                                            |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem başarılıysa - true, değilse - false.               |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrder()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

DeleteOrderLong

Bekleyen uzun Limit/Stop emrini siler.

```
virtual bool  DeleteOrderLong()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Bekleyen uzun emri (alış Limit/Stop) siler (CTrade sınıfının OrderDelete(...) yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Uzun limit/stop emrini sil                                       |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem başarılıysa - true, değilse - false.               |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrderLong()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

DeleteOrderShort

Bekleyen kısa (satış) Limit/Stop emrini siler

```
virtual bool  DeleteOrderShort()
```

Dönüş değeri

İşlem uygulanmışsa 'true', aksi durumda 'false'.

Not

Bekleyen kısa Limit/Stop emrini siler (CTrade sınıfının OrderDelete(...) yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Kısa limit/stop emrini sil                                       |
//| INPUT:  yok.                                                     |
//| OUTPUT: işlem başarılıysa - true, değilse - false.               |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
bool CExpert::DeleteOrderShort()
  {
   return(m_trade.OrderDelete(m_order.Ticket()));
  }
```

LotOpenLong

Uzun (alış) işlemin hacim değerini alır.

```
double  LotOpenLong(
   double    price,    // fiyat
   double    sl        // Stop Loss
   )
```

Parametreler

price

[in] Fiyat.

sl

[in] Stop Loss (zararı durdur) fiyatı.

Dönüş değeri

Alım işleminin hacmi (lot olarak).

Not

Alım işleminin hacim değerini alır (para yönetimi nesnesinin CheckOpenLong(...) yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Uzun pozisyon için lot değeri alma yöntemi.                      |
//| INPUT:  price - fiyat,                                           |
//|         sl    - stop loss.                                       |
//| OUTPUT: açılış lotu.                                             |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
double CExpert::LotOpenLong(double price,double sl)
  {
   return(m_money.CheckOpenLong(price,sl));
  }
```

LotOpenShort

Kısa (satış) işlemin hacim değerini alır.

```
double  LotOpenShort(
   double    price,   // fiyat
   double    sl       // Stop Loss
   )
```

Parametreler

price

[in] Fiyat.

sl

[in] Stop Loss fiyatı.

Dönüş değeri

Satış işleminin hacmi (lot olarak).

Not

Satış işleminin hacim değerini alır (para yönetimi nesnesinin CheckOpenShort(...) yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Kısa pozisyon için lot değeri alma yöntemi.                      |
//| INPUT:  price - fiyat,                                           |
//|         sl    - stop loss.                                       |
//| OUTPUT: açılış lotu.                                             |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
double CExpert::LotOpenShort(double price,double sl)
  {
   return(m_money.CheckOpenShort(price,sl));
  }
```

LotReverse

Pozisyonun ters çevrilmesinde kullanılan işlem hacmi değerini alır.

```
double  LotReverse(
   double    sl       // Stop Loss
   )
```

Parametreler

sl

[in] Stop Loss (zararı durdur) fiyatı.

Dönüş değeri

Ters çevirme işleminin hacmi (lot olarak).

Not

Ters çevirme işleminin hacim değerini alır (para yönetimi nesnesinin CheckReverse(...) yöntemi).

Uygulama

```
//+------------------------------------------------------------------+
//| Ters pozisyon için lot değeri alma yöntemi.                      |
//| INPUT:  sl - stop loss.                                          |
//| OUTPUT: açılış lotu.                                             |
//| REMARK: yok.                                                     |
//+------------------------------------------------------------------+
double CExpert::LotReverse(double sl)
  {
   return(m_money.CheckReverse(GetPointer(m_position),sl));
  }
```

PrepareHistoryDate

Geçmiş takibi için başlangıç tarihi oluşturur.

```
void  PrepareHistoryDate()
```

Not

İşlem geçmişi takip periyodu, ayın başına ayarlıdır (ve bir günden az olamaz).

HistoryPoint

İşlem geçmişi için bir kontrol noktası oluşturur (pozisyonların, emirlerin, ve geçmiş işlemlerin sayısını kaydeder).

```
void  HistoryPoint(
   bool   from_check_trade=false   // bayrak
   )
```

Parametreler

from_check_trade=false

[in] Tekrarı engellemek için bir bayrak.

Not

Pozisyonların, emirlerin, ve geçmiş işlemlerin sayısını kaydeder.

CheckTradeState

Mevcut durumu kaydedilmiş olanla karşılaştırır ve ilgili olay işleyicisini çağırır.

```
bool  CheckTradeState()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

pozisyonların sayısını, emirleri, işlemleri ve geçmiş emirleri, HistoryPoint() yöntemiyle kaydedilmiş olan değerlerle karşılaştırarak denetler. Alım-satım geçmişi değişmişse, karşılık gelen sanal olay işleyicisine çağrı yapar.

WaitEvent

Olay bekleme bayrağını ayarlar.

```
void  WaitEvent(
   ENUM_TRADE_EVENTS    event          // bayrak
   )
```

Parametreler

event

[in] Ayarlanacak olay bayrağı (ENUM_TRADE_EVENTS sayımının değerlerinden biri).

Dönüş değeri

Yok.

Olay Bayrakları

```
//--- beklenen olayların bayrakları
enum ENUM_TRADE_EVENTS
  {
   TRADE_EVENT_NO_EVENT               =0,         // beklenen olay yok
   TRADE_EVENT_POSITION_OPEN          =0x1,       // "pozisyon açılışı" olayının bekler
   TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,        // "pozisyon değişimi" olayının bekle
   TRADE_EVENT_POSITION_MODIFY        =0x4,       // "stop emrinin değişimi" olayının b
   TRADE_EVENT_POSITION_CLOSE         =0x8,       // "pozisyon kapanışı" olayının bekle
   TRADE_EVENT_POSITION_STOP_TAKE     =0x10,      // "pozisyonun stop emrinin tetiklenm
   TRADE_EVENT_ORDER_PLACE            =0x20,      // "bekleyen emir yerleştirme" olayın
   TRADE_EVENT_ORDER_MODIFY           =0x40,      // "bekleyen emrin değiştirilmesi" ol
   TRADE_EVENT_ORDER_DELETE           =0x80,      // "bekleyen emrin silinmesi" olayını
   TRADE_EVENT_ORDER_TRIGGER          =0x100      // "bekleen emrin tetiklenmesi" olayı
  };
```

NoWaitEvent

Beklenen olay bayrağını sıfırlar.

```
void  NoWaitEvent(
   ENUM_TRADE_EVENTS    event          // bayrak
   )
```

Parametreler

event

[in] Sıfırlanacak olay bayrağı (ENUM_TRADE_EVENTS sayımının değerlerinden biri).

Dönüş değeri

Yok.

Olay Bayrakları

```
//--- beklenen olayların bayrakları
enum ENUM_TRADE_EVENTS
  {
   TRADE_EVENT_NO_EVENT               =0,         // beklenen olay yok
   TRADE_EVENT_POSITION_OPEN          =0x1,       // "pozisyon açılışı" olayının bekler
   TRADE_EVENT_POSITION_VOLUME_CHANGE=0x2,        // "pozisyon değişimi" olayının bekle
   TRADE_EVENT_POSITION_MODIFY        =0x4,       // "stop emrinin değişimi" olayının b
   TRADE_EVENT_POSITION_CLOSE         =0x8,       // "pozisyon kapanışı" olayının bekle
   TRADE_EVENT_POSITION_STOP_TAKE     =0x10,      // "pozisyonun stop emrinin tetiklenm
   TRADE_EVENT_ORDER_PLACE            =0x20,      // "bekleyen emir yerleştirme" olayın
   TRADE_EVENT_ORDER_MODIFY           =0x40,      // "bekleyen emrin değiştirilmesi" ol
   TRADE_EVENT_ORDER_DELETE           =0x80,      // "bekleyen emrin silinmesi" olayını
   TRADE_EVENT_ORDER_TRIGGER          =0x100      // "bekleen emrin tetiklenmesi" olayı
  };
```

TradeEventPositionStopTake

"Pozisyonun Stop Loss/Take Profit seviyeleri tetiklendi" olayının işleyicisi.

```
virtual bool  TradeEventPositionStopTake()
```

Dönüş değeri

CExpert sınıfının yöntemi herhangi bir şey yapmaz ve daima 'true' döüşü yapar.

TradeEventOrderTriggered

"Bekleyen emir tetiklendi" olayının işleyicisi.

```
virtual bool  TradeEventOrderTriggered()
```

Dönüş değeri

CExpert sınıfının yöntemi herhangi bir şey yapmaz ve daima 'true' döüşü yapar.

TradeEventPositionOpened

"Pozisyon açıldı" olayının işleyicisi.

```
virtual bool  TradeEventPositionOpened()
```

Dönüş değeri

CExpert sınıfının yöntemi herhangi bir şey yapmaz ve daima 'true' döüşü yapar.

TradeEventPositionVolumeChanged

"Pozisyon hacmi değişti" olayının işleyicisi.

```
virtual bool  TradeEventPositionVolumeChanged()
```

Dönüş değeri

CExpert sınıfının yöntemi herhangi bir şey yapmaz ve daima 'true' döüşü yapar.

TradeEventPositionModified

"Pozisyon değiştirildi" olayının işleyicisi.

```
virtual bool  TradeEventPositionModified()
```

Dönüş değeri

CExpert sınıfının yöntemi herhangi bir şey yapmaz ve daima 'true' döüşü yapar.

TradeEventPositionClosed

"Pozisyon kapandı" olayının işleyicisi.

```
virtual bool  TradeEventPositionClosed()
```

Dönüş değeri

CExpert sınıfının yöntemi herhangi bir şey yapmaz ve daima 'true' döüşü yapar.

TradeEventOrderPlaced

"Bekleyen emir oluşturuldu" olayının işleyicisi.

```
virtual bool  TradeEventOrderPlaced()
```

Dönüş değeri

CExpert sınıfının yöntemi herhangi bir şey yapmaz ve daima 'true' döüşü yapar.

TradeEventOrderModified

"Bekleyen emir değiştirildi" olayının işleyicisi.

```
virtual bool  TradeEventOrderModified()
```

Dönüş değeri

CExpert sınıfının yöntemi herhangi bir şey yapmaz ve daima 'true' döüşü yapar.

TradeEventOrderDeleted

"Bekleyen emir silindi" olayının işleyicisi.

```
virtual bool  TradeEventOrderDeleted()
```

Dönüş değeri

CExpert sınıfının yöntemi herhangi bir şey yapmaz ve daima 'true' döüşü yapar.

TradeEventNotIdentified

Tanımlanmayan olay için olay işleyici.

```
virtual bool  TradeEventNotIdentified()
```

Dönüş değeri

CExpert sınıfının yöntemi herhangi bir şey yapmaz ve daima 'true' döüşü yapar.

Not

Birkaç alım-satım olayının birlikte oluşabileceğini not ediniz. Böyle bir durumda olayları tanımlamak zorlaşacaktır.

TimeframeAdd

İzlenecek zaman-dilimi ekler.

```
void  TimeframeAdd(
   ENUM_TIMEFRAMES    period         // zaman-dilimi
   )
```

Parametreler

period

[in] Zaman-dilimi (ENUM_TIMEFRAMES sayımının değerleri).

Dönüş değeri

Yok.

TimeframesFlags

Yeni çubukla birlikte zaman-dilimi bayrağını alır

```
int  TimeframesFlags(
   MqlDateTime&     time          // zaman değişkeni
   )
```

Parametreler

time

[in] Referansla geçirilen MqlDateTime tipli yeni zaman değişkeni.

Dönüş değeri

Zaman-dilimini gösteren bayrağı alır.

CExpertSignal

CExpertSignal, alım-satım sinyalleri için temel sınıftır, arayüz sağlamak dışında herhangi bir eylem gerçekleştirmez (CheckReverseLong() ve CheckReverseShort() yöntemleri hariç).

Nasıl kullanılır:

1. Alım-satım sinyalleri için bir algoritma hazırlayın;
2. CExpertSignal sınıfının soyundan gelen, kendi alım-satım sınıfınızı oluşturun;
3. Sınıfınızdaki sanal yöntemleri kendi algoritmanızı kullanarak yeniden tanımlayın.

Expert\Signal\ klasöründe alım-satım sinyalleri sınıflarına örnekler bulabilirsiniz.

Açıklama

CExpertSignal, alım-satım sinyallerinin uygulanmaları için bir temel sınıftır.

Bildirim

```
class CExpertSignal : public CExpertBase
```

Başlık

```
#include <Expert\ExpertSignal.mqh>
```

Kalıtım hiyerarşisi

CObject
  CExpertBase
    CExpertSignal

İlk nesil

CSignalAC, CSignalAMA, CSignalAO, CSignalBearsPower, CSignalBullsPower, CSignalCCI, CSignalDeM, CSignalDEMA, CSignalEnvelopes, CSignalFrAMA, CSignalRSI, CSignalRVI, CSignalSAR, CSignalStoch, CSignalTEMA, CSignalTriX, CSignalWPR

Sınıf Yöntemleri

Başlatma	
virtual InitIndicators	Göstergeleri ve zaman serilerini başlatır
virtual ValidationSettings	Ayarları denetler
virtual AddFilter	Birleştirilmiş sinyale bir filtre ekler
Korunan Veriye Erişim	
BasePrice	Temel fiyat seviyesini ayarlar
UsedSeries	Kullanılan zaman serilerinin bayraklarını alır
Parametrelerin Ayarlanması	

Başlatma	
Weight	"Weight" parametresinin değerini ayarlar
PatternsUsage	"PatternsUsage" parametresinin değerini ayarlar
General	"General" parametresinin değerini ayarlar
Ignore	"Ignore" parametresinin değerini ayarlar
Invert	"Invert" parametresinin değerini ayarlar
ThresholdOpen	"ThresholdOpen" parametresinin değerini ayarlar
ThresholdClose	"ThresholdClose" parametresinin değerini ayarlar
PriceLevel	"PriceLevel" parametresinin değerini ayarlar
StopLevel	"StopLevel" parametresinin değerini ayarlar
TakeLevel	"TakeLevel" parametresinin değerini ayarlar
Expiration	"Expiration" parametresinin değerini ayarlar
Magic	"Magic" parametresinin değerini ayarlar
Alım-Satım Koşullarının Denetimi	
virtual CheckOpenLong	Uzun pozisyon açma koşullarını denetler
virtual CheckCloseLong	Uzun pozisyon kapama koşullarını denetler
virtual CheckOpenShort	Kısa pozisyon açma koşullarını denetler
virtual CheckCloseShort	Kısa pozisyon kapama koşullarını denetler
virtual CheckReverseLong	Uzun pozisyon için ters çevirme koşullarını denetler
virtual CheckReverseShort	Kıza pozisyon için ters çevirme koşullarını denetler
Alım-Satım Parametrelerinin Ayarlanması	
virtual OpenLongParams	Uzun pozisyon açma parametrelerini ayarlar
virtual OpenShortParams	Kısa pozisyon açma parametrelerini ayarlar
virtual CloseLongParams	Uzun pozisyon kapama parametrelerini ayarlar
virtual CloseShortParams	Kısa pozisyon kapama parametrelerini ayarlar
Bekleyen Emirlerin Değiştirilme Koşullarının Denetlenmesi	
virtual CheckTrailingOrderLong	Bekleyen alış emrinin parametrelerinin değişimi için koşulları denetler
virtual CheckTrailingOrderShort	Bekleyen satış emrinin parametrelerinin değişimi için koşulları denetler

Başlatma	
Piyasa Emirlerinin Biçimini Denetlemek için Yöntemler	
virtual LongCondition	Alış koşulları denetiminin sonucunu alır
virtual ShortCondition	Alış koşulları denetiminin sonucunu alır
virtual Direction	Fiyatın ağırlıklı yönünü alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries

BasePrice

Temel fiyat seviyesini ayarlar.

```
void  BasePrice(
   double    value          // yeni değer
   )
```

Parametreler

value

[in] Temel fiyat seviyesi için yeni değer.

Dönüş değeri

Yok.

UsedSeries

Kullanılan zaman-serilerinin bayraklarını alır.

```
int  UsedSeries()
```

Dönüş değeri

Kullanılan serilerinin bayraklarına (sembol/periyot değerleri kullanılan sembol/periyot değerlerine karşılık geliyorsa), aksi durumda 0 değerine dönüş yapar.

Weight

"Weight" parametresinin değerini ayarlar.

```
void  Weight(
   double    value         // yeni değer
   )
```

Parametreler

value

[in] "Weight" parametresinin yeni değeri.

Dönüş değeri

Yok.

PatternUsage

"PatternsUsage" parametresinin değerini ayarlar.

```
void  PatternUsage(
   double    value          // yeni değer
   )
```

Parametreler

value

[in] "PatternsUsage" için yeni değer.

Dönüş değeri

Yok.

General

"General" parametresinin yeni değerini ayarlar.

```
void  General(
   int     value          // yeni değer
   )
```

Parametreler

value

[in] "General" parametresinin yeni değeri.

Dönüş değeri

Yok.

Ignore

"Ignore" parametresinin değerini ayarlar.

```
void  Ignore(
   long     value          // yeni değer
   )
```

Parametreler

value

[in] "Ignore" parametresinin yeni değeri.

Dönüş değeri

Yok.

Invert

"Invert" parametresinin değerini ayarlar

```
void  Invert(
   long     value          // yeni değer
   )
```

Parametreler

value

[in] "Invert" parametresinin yeni değeri.

Dönüş değeri

Yok.

ThresholdOpen

"ThresholdOpen" parametresinin değerini ayarlar.

```
void  ThresholdOpen(
   long     value          // yeni değer
   )
```

Parametreler

value

[in] "ThresholdOpen" parametresinin yeni değeri.

Dönüş değeri

Yok.

Not

"ThresholdOpen" parametresi 0 - 100 arası değerler alır. Pozisyon açma koşulları değerlendirilirken (oylanırken) kullanılır.

ThresholdClose

"ThresholdClose" parametresinin değerini ayarlar.

```
void  ThresholdOpen(
   long     value          // yeni değer
   )
```

Parametreler

value

[in] "ThresholdClose" parametresinin yeni değeri.

Dönüş değeri

Yok.

Not

"ThresholdClose" parametresi 0 - 100 arası değerler alır. Pozisyon kapama koşulları değerlendirilirken (oylanırken) kullanılır.

PriceLevel

"PriceLevel" parametresinin değerini ayarlar.

```
void  PriceLevel(
   double    value         // yeni değer
   )
```

Parametreler

value

[in] "PriceLevel" parametresinin yeni değeri.

Dönüş değeri

Yok.

Not

"PriceLevel" değeri fiyat seviyesi birimlerinin cinsinden tanımlanır. Fiyat seviyesi biriminin sayısal değeri PriceLevelUnit() yöntemi ile alınır. "PriceLevel", açılış fiyatını temel fiyata göre tanımlamak için kullanılır.

StopLevel

"StopLevel" parametresinin değerini ayarlar.

```
void  StopLevel(
   double    value          // yeni değer
   )
```

Parametreler

value

[in] "StopLevel" parametresinin yeni değeri.

Dönüş değeri

Yok.

Not

"StopLevel" değeri fiyat seviyesi birimlerinin cinsinden tanımlanır. Fiyat seviyesi biriminin sayısal değeri PriceLevelUnit() yöntemi ile alınır. "StopLevel", Stop Loss fiyatını açılış fiyatına göre tanımlamak için kullanılır.

TakeLevel

"TakeLevel" parametresinin değerini ayarlar.

```
void  TakeLevel(
   double    value         // yeni değer
   )
```

Parametreler

value

[in] "TakeLevel" parametresinin yeni değeri.

Dönüş değeri

Yok.

Not

"TakeLevel" değeri fiyat seviyesi birimlerinin cinsinden tanımlanır. Fiyat seviyesi biriminin sayısal değeri PriceLevelUnit() yöntemi ile alınır. "TakeLevel", Take Profit fiyatını açılış fiyatına göre tanımlamak için kullanılır.

Expiration

"Expiration" (zaman-aşımı) parametresinin değerini ayarlar.

```
void  Expiration(
   int     value          // yeni değer
   )
```

Parametreler

value

[in] "Expiration" için yeni değer.

Dönüş değeri

Yok.

Not

"Expiration" parametresinin değeri çubuk bazında tanımlanır. Bu, bekleyen emirlerle işlem yaparken zaman-aşımı değeri olarak kullanılır.

Magic

"Magic" parametresinin değerini ayarlar.

```
void  Magic(
   int    value         // yeni değer
   )
```

Parametreler

value

[in] "Magic" (Uzman Danışman tanımlayıcısı) parametresinin yeni değeri.

Dönüş değeri

Yok.

ValidationSettings

Ayarları denetler.

```
virtual bool  ValidationSettings()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

InitIndicators

Tüm göstergeleri ve zaman-serilerini başlatır.

```
virtual bool  InitIndicators(
   CIndicators*  indicators    // işaretçi
   )
```

Parametreler

indicators

[in] Göstergelerin ve zaman-serilerinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Nesne, başlatma sırasında tanımlanan sembol veya zaman diliminden farklı sembol veya zaman dilimi kullanıyorsa, zaman-serileri başlatılır.

AddFilter

Birleştirilmiş sinyale bir filtre ekler.

```
virtual bool  AddFilter(
   CExpertSignal*  filter    // işaretçi
   )
```

Parametreler

indicators

[in] Filtre nesnesinin işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CheckOpenLong

Uzun pozisyon açma koşullarını denetler.

```
virtual bool  CheckOpenLong(
   double&    price,          // fiyat
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // zaman-aşımı
   )
```

Parametreler

price

[in][out] Referansla geçirilen fiyat değişkeni.

sl

[in][out] Referansla geçirilen Stop Loss fiyatı değişkeni.

tp

[in][out] Referansla geçirilen Take Profit fiyatı değişkeni.

expiration

[in][out] Referansla geçirilen zaman-aşımı süresi değişkeni.

Dönüş değeri

Koşul sağlanmışsa 'true', aksi durumda 'false'.

CheckOpenShort

Kısa pozisyon açma koşullarını denetler.

```
virtual bool  CheckOpenShort(
   double&    price,          // fiyat
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // zaman-aşımı
   )
```

Parametreler

price

[in][out] Referansla geçirilen fiyat değişkeni.

sl

[in][out] Referansla geçirilen Stop Loss fiyatı değişkeni.

tp

[in][out] Referansla geçirilen Take Profit fiyatı değişkeni.

expiration

[in][out] Referansla geçirilen zaman-aşımı süresi değişkeni.

Dönüş değeri

Koşul sağlanmışsa 'true', aksi durumda 'false'.

OpenLongParams

Uzun pozisyon açmak için kullanılan parametreleri ayarlar.

```
virtual bool  OpenLongParams(
   double&    price,          // fiyat
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // zaman-aşımı
   )
```

Parametreler

price

[in][out] Referansla geçirilen fiyat değişkeni.

sl

[in][out] Referansla geçirilen Stop Loss fiyatı değişkeni.

tp

[in][out] Referansla geçirilen Take Profit fiyatı değişkeni.

expiration

[in][out] Referansla geçirilen zaman-aşımı süresi değişkeni.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OpenShortParams

Kısa pozisyon açmak için kullanılan parametreleri ayarlar.

```
virtual bool  OpenShortParams(
   double&    price,          // fiyat
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // zaman-aşımı
   )
```

Parametreler

price

[in][out] Referansla geçirilen fiyat değişkeni.

sl

[in][out] Referansla geçirilen Stop Loss fiyatı değişkeni.

tp

[in][out] Referansla geçirilen Take Profit fiyatı değişkeni.

expiration

[in][out] Referansla geçirilen zaman-aşımı süresi değişkeni.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CheckCloseLong

Uzun pozisyon kapama koşullarını denetler.

```
virtual bool  CheckCloseLong(
   double&  price      // fiyat
   )
```

Parametreler

price

[in][out] Referansla geçirilen kapanış fiyatı değişkeni.

Dönüş değeri

Koşul sağlanmışsa 'true', aksi durumda 'false'.

CheckCloseShort

Kısa pozisyon kapama koşullarını denetler.

```
virtual bool  CheckCloseShort(
   double&  price      // fiyat
   )
```

Parametreler

price

[in][out] Referansla geçirilen kapanış fiyatı değişkeni.

Dönüş değeri

Koşul sağlanmışsa 'true', aksi durumda 'false'.

CloseLongParams

Uzun pozisyon kapamak için kullanılan parametreleri ayarlar.

```
virtual bool  CloseLongParams(
   double&    price          // fiyat
   )
```

Parametreler

price

[in][out] Referansla geçirilen kapanış fiyatı değişkeni.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CloseShortParams

Kısa pozisyon kapamak için kullanılan parametreleri ayarlar.

```
virtual bool  CloseShortParams(
   double&    price          // fiyat
   )
```

Parametreler

price

[in][out] Referansla geçirilen kapanış fiyatı değişkeni.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CheckReverseLong

Uzun pozisyon için ters çevirme koşullarını denetler.

```
virtual bool  CheckReverseLong(
   double&    price,          // fiyat
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // zaman-aşımı
   )
```

Parametreler

price

[in][out] Referansla geçirilen fiyat değişkeni.

sl

[in][out] Referansla geçirilen Stop Loss fiyatı değişkeni.

tp

[in][out] Referansla geçirilen Take Profit fiyatı değişkeni.

expiration

[in][out] Referansla geçirilen zaman-aşımı süresi değişkeni.

Dönüş değeri

Koşul sağlanmışsa 'true', aksi durumda 'false'.

CheckReverseShort

Kıza pozisyon için ters çevirme koşullarını denetler.

```
virtual bool  CheckReverseShort(
   double&    price,          // fiyat
   double&    sl,             // Stop Loss
   double&    tp,             // Take Profit
   datetime&  expiration      // zaman-aşımı
   )
```

Parametreler

price

[in][out] Referansla geçirilen ters çevirme fiyatı değişkeni.

sl

[in][out] Referansla geçirilen Stop Loss fiyatı değişkeni.

tp

[in][out] Referansla geçirilen Take Profit fiyatı değişkeni.

expiration

[in][out] Referansla geçirilen zaman-aşımı süresi değişkeni.

Dönüş değeri

Koşul sağlanmışsa 'true', aksi durumda 'false'.

CheckTrailingOrderLong

Bekleyen alış emrinin parametrelerinin değişimi için koşulları denetler.

```
virtual bool  CheckTrailingOrderLong(
   COrderInfo*     order,          // emir
   double&         price           // fiyat
   )
```

Parametreler

order

[in] COrderInfo sınıf nesnesinin işaretçisi.

price

[in][out] Stop Loss fiyatının değişkeni.

Dönüş değeri

Koşul sağlanmışsa 'true', aksi durumda 'false'.

CheckTrailingOrderShort

Bekleyen satış emrinin parametrelerinin değişimi için koşulları denetler.

```
virtual bool  CheckTrailingOrderShort(
   COrderInfo*     order,          // emir
   double&         price           // fiyat
   )
```

Parametreler

order

[in] COrderInfo sınıf nesnesinin işaretçisi.

price

[in][out] Stop Loss fiyatının değişkeni.

Dönüş değeri

Koşul sağlanmışsa 'true', aksi durumda 'false'.

LongCondition

Uzun pozisyon açma koşullarını denetler.

```
virtual int  LongCondition()
```

Dönüş değeri

Koşullar sağlandığında, sinyal gücüne bağlı olarak 1-100 arasında bir değere dönüş yapar, pozisyon açabilmek için bir sinyal yoksa 0 dönüşü yapar.

Not

Temel sınıfın LongCondition() yönteminin, uzun pozisyon koşullarını denetleme gibi bir yetisi yoktur ve her zaman 0 dönüşü yapar.

ShortCondition

Kısa pozisyon açma koşullarını denetler.

```
virtual int  ShortCondition()
```

Dönüş değeri

Koşullar sağlandığında, sinyal gücüne bağlı olarak 1-100 arasında bir değere dönüş yapar, pozisyon açabilmek için bir sinyal yoksa 0 dönüşü yapar.

Not

Temel sınıfın ShortCondition() yönteminin, kısa pozisyon koşullarını denetleme gibi bir yetisi yoktur ve her zaman 0 dönüşü yapar.

Direction

"Ağırlıklı" fiyat yönünün değerine dönüş yapar.

```
virtual double  Direction()
```

Dönüş değeri

Yukarı yönlü ağırlık için >0, aşağı yönlü ağırlık için ise <0 değerlerine dönüş yapar. Değerin kesinliği sinyalin "gücüne" bağlıdır.

Not

Filtre kullanılması durumunda sonuç filtrelere bağlı olarak değişir.

CExpertTrailing

CExpertTrailing, iz-sürme algoritmaları için bir temel sınıftır, arayüz sağlama dışında bir eylem gerçekleştirmez.

Nasıl kullanılır:

1. Bir iz-sürme algoritması hazırlayın;
2. CExpertTrailing sınıfının soyundan gelen bir iz-sürme sınıfı oluşturun;
3. Sınıfınızdaki sanal yöntemleri kendi algoritmanızı kullanarak yeniden tanımlayın.

Expert\Trailing\ klasöründe iz-sürme sınıflarının örneklerini bulabilirsiniz.

Açıklama

CExpertTrailing, iz-sürme algoritmalarıyla yapılan uygulamalar için bir temel sınıftır.

Bildirim

```
class CExpertTrailing : public CExpertBase
```

Başlık

```
#include <Expert\ExpertTrailing.mqh>
```

Kalıtım hiyerarşisi

CObject

CExpertBase

CExpertTrailing

İlk nesil

CTrailingFixedPips, CTrailingMA, CTrailingNone, CTrailingPSAR

Sınıf Yöntemleri

Stop Emirleri için Değiştirme Koşullarının Denetimi	
virtual CheckTrailingStopLong	Uzun pozisyon parametrelerinin değişimi için koşulları denetler
virtual CheckTrailingStopShort	Kısa pozisyon parametrelerinin değişimi için koşulları denetler

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries, InitIndicators

CheckTrailingStopLong

Uzun pozisyon parametrelerinin değişimi için koşulları denetler.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // işaretçi
   double&         sl,            // Stop Loss
   double&         tp             // Take Profit
   )
```

Parametreler

position

[in] CPositionInfo isimli sınıf örneğinin işaretçisi.

sl

[in][out] Referansla geçirilen Stop Loss fiyatı değişkeni.

tp

[in][out] Referansla geçirilen Take Profit fiyatı değişkeni.

Dönüş değeri

Koşul sağlanmışsa 'true', aksi durumda 'false'.

Not

Temel sınıfın CheckTrailingStopLong(...) daima 'false' dönüşü yapar.

CheckTrailingStopShort

Kısa pozisyon parametrelerinin değişimi için koşulları denetler.

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // işaretçi
   double&         sl,           // Stop Loss
   double&         tp            // Take Profit
   )
```

Parametreler

position

[in] CPositionInfo isimli sınıf örneğinin işaretçisi.

sl

[in][out] Referansla geçirilen Stop Loss fiyatı değişkeni.

tp

[in][out] Referansla geçirilen Take Profit fiyatı değişkeni.

Dönüş değeri

Koşul sağlanmışsa 'true', aksi durumda 'false'.

Not

Temel sınıfın CheckTrailingStopShort(...) daima 'false' dönüşü yapar.

CExpertMoney

CExpertMoney, para ve risk yönetimi algoritmaları için temel sınıftır.

Açıklama

CExpertMoney, para ve risk yönetim sınıflarının uygulanması için bir temel sınıftır.

Bildirim

```
class CExpertMoney : public CObject
```

Başlık

```
#include <Expert\ExpertMoney.mqh>
```

Kalıtım hiyerarşisi

CObject

CExpertBase

CExpertMoney

İlk nesil

CMoneyFixedLot, CMoneyFixedMargin, CMoneyFixedRisk, CMoneyNone, CMoneySizeOptimized

Sınıf Yöntemleri

Korunan Veriye Erişim	
Percent	"Risk yüzdesi" parametresini ayarlar
Başlatma	
virtual ValidationSettings	Ayarları denetler
Alım-Satım Koşullarının Denetimi	
virtual CheckOpenLong	Uzun pozisyon için hacim değerini alır
virtual CheckOpenShort	Kısa pozisyon için hacim değerini alır
virtual CheckReverse	Pozisyonun tersi için hacim değerini alır
virtual CheckClose	Açık pozisyonların kapatılma koşullarını denetler

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Percent

"Risk yüzdesi" parametresini ayarlar.

```
void  Percent(
   double  percent      // risk yüzdesi
   )
```

Parametreler

percent

[in] Risk yüzdesi.

Dönüş değeri

Yok.

ValidationSettings

Ayarları denetler.

```
virtual bool  ValidationSettings()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Temel sınıfın ValidationSettings() yöntemi her zaman 'true' dönüşü yapar.

CheckOpenLong

Uzun pozisyon için hacim değerini alır.

```
virtual double  CheckOpenLong(
   double  price,     // fiyat
   double  sl         // Stop Loss
   )
```

Parametreler

price

[in] Uzun pozisyonun açılış fiyatı.

sl

[in] Stop Loss (zararı durdur) fiyatı.

Dönüş değeri

Uzun pozisyon için işlem hacmi değeri.

CheckOpenShort

Kısa pozisyon için hacim değerini alır.

```
virtual double  CheckOpenShort(
   double  price,     // fiyat
   double  sl         // Stop Loss
   )
```

Parametreler

price

[in] Kısa pozisyonun açılış fiyatı.

sl

[in] Stop Loss (zararı durdur) fiyatı.

Dönüş değeri

Kısa pozisyon için işlem hacmi değeri.

CheckReverse

Pozisyonun tersi için hacim değerini alır.

```
virtual double  CheckReverse(
   CPositionInfo*  position,     // işaretçi
   double          sl            // Stop Loss
   )
```

Parametreler

position

[in] CPositionInfo sınıf nesnesinin işaretçisi.

sl

[in] Stop Loss (zararı durdur) fiyatı.

Dönüş değeri

Pozisyonun tersi için işlem hacmi değeri.

CheckClose

Açık pozisyonların kapatılma koşullarını denetler.

```
virtual double  CheckClose()
```

Dönüş değeri

Koşul sağlanmışsa 'true', aksi durumda 'false'.

Alım-Satım Sinyallerinin Modülleri

Müşteri terminalinin standart dağıtımı "MQL5 Sihirbazı" için bir dizi kullanıma-hazır sinyal modülü içerir. MQL5 Sihirbazında bir Expert Advisor oluştururken, alım satım sinyallerinin tüm kombinasyonlarını (64 'e kadar) kullanabilirsiniz. Alım-satım işlemi için son karar, dahil edilen her bir modülden elde edilen sinyallerin karmaşık analizine göre alınır. Alim-satım kararlarının alınma mekanizması aşağıda ayrıntılarıyla açıklanmıştır.

Standart dağıtım şu sinyal modüllerini içerir:

- Accelerator Oscillator Göstergesinin Sinyalleri
- Adaptive Moving Average Göstergesinin Sinyalleri
- Awesome Oscillator göstergesinin sinyalleri
- Bears Power Osilatörünün Sinyalleri
- Bulls Power Osilatörünün Sinyalleri
- Commodity Channel Index Osilatörünün Sinyalleri
- DeMarker Osilatörünün Sinyalleri
- Double Exponential Moving Average Göstergesinin Sinyalleri
- Envelopes göstergesinin Sinyalleri
- Fractal Adaptive Moving Average Göstergesinin Sinyalleri
- Intraday Time Filter Sinyalleri
- MACD Osilatörünün Sinyalleri
- Moving Average Göstergesinin Sinyalleri
- Parabolic SAR Göstergesinin Sinyalleri
- Relative Strength Index Osilatörünün Sinyalleri
- Relative Vigor Index Osilatörünün Sinyalleri
- Stochastic Osilatörünün Sinyalleri
- Triple Exponential Average Osilatörünün Sinyalleri
- Triple Exponential Moving Average Göstergesinin Sinyalleri
- Oscillator Williams Percent Range Osilatörünün Sinyalleri

Sinyal Modüllerine Dayanan Alım-Satım Karar Mekanizması

Alım-satım karar mekanizması aşağıda verilen temel prensipler çerçevesinde açıklanabilir:

- Her bir sinyal modülü kendi piyasa modeline sahiptir (fiyat ve gösterge değerlerinin belirli kombinasyonlar).
- Her piyasa modeli 1-100 arasında değişen anlamlılık değerine sahiptir. Anlamlılık değeri ne kadar yüksekse, model o kadar güçlüdür.
- Her bir model fiyat hareketinin yönüne dair tahminler oluşturur.
- Bir modülün tahmini, eklenen modeller için yapılan aramanın sonucudur ve -100 ile 100 arasında bir değer olarak çıktılanır. İşaret, tahmini hareketin yönünü belirtir (negatif işaret fiyatın düşeceğini, pzitif işaret ise fiyatın yükseleceğini ifade eder). Mutlak değer, elde edilen en iyi modelin gücüne karşılık gelir.

- Her bir modülün tahmini, modül içinde belirlenen 0-1 aralığındaki bir ağırlık katsayısı ile "oylamaya" gönderilir.
- Oylamanın sunucu, -100 ile 100 arasında değişen bir sayıdır. Bu sayının işareti tahmini hareketin yönünü belirtirken, mutlak değeri ise sinyal gücünü temsil eder. Tüm sinyal modüllerinin tahminlerinin ağırlıklı ortalaması şeklinde hesaplanır.

Oluşturulan her Uzman Danışman değiştirilebilir iki ayara sahiptir — 0 ile 100 arasında değerler alabilen pozisyon açma ve kapama eşikleri (ThresholdOpen ve ThresholdClose). Son sinyalin gücü eşik seviyesini aştığında, sinyal işaretine karşılık gelen alım-satım işlemi gerçekleştirilir.

Örnekler

Şu eşik seviyelerine sahip bir Uzman Danışman ele alalım: ThresholdOpen=20 ve ThresholdClose=90. Alım-satım işlemleri karar mekanizmasında iki sinyal modülü bulunsun : 0.4 ağırlığa sahip MA modülü ve 0.8 ağırlığa sahip Stochastic modülü. Şimdi, elde edilebilecek sinyallerin iki çeşidini inceleyelim:

Çeşit 1.

Fiyat serisi, MA göstergesini aşağıdan yukarı doğru kesmiş olun. Bu durum, MA modülü içinde uygulanan piyasa modellerinin birine karşılık gelmektedir. Söz konusu model, fiyatların yükselişini öngörür. Anlamlılık değeri 100 'e eşittir. Bu sırada Stochastic osilatör de aşağı dönmüş olsun ve fiyat serisi ile bir ayrışma göstersin. Bu durum, Stochastic modülü içinde uygulanan piyasa modellerinin birine karşılık gelmektedir. Söz konusu model, fiyatların düşüşünü öngörür. Anlamlılık değeri 80 'e eşittir.

Şimdi nihai "oylamanın" sonucunu hesaplayabiliriz. MA modülünden elde edilen oran, 0.4 * 100 = 40 şeklinde hesaplanır. Stochastic modülünden elde edilen oran, 0.8 * (-80) = -64 şeklinde hesaplanır. Son değer, bu iki oranın aritmetik ortalamasıyla bulunur: (40 - 64)/2 = -12. Oylamanın sonucu, izafi gücü 12 'ye eşit olan olan satış sinyalidir. 20 değerine sahip işlem eşiğine ulaşılamamıştır. Bu sebeple Herhangi bir alım-satım işlemi gerçekleştirilmez.

Çeşit 2.

Fiyat serisi, MA göstergesini yukarıdan aşağı doğru kesmiş olun. Bu durum, MA modülü içinde uygulanan piyasa modellerinin birine karşılık gelir. Söz konusu model, fiyatların yükselişini öngörür. Anlamlılık değeri 10 'a eşittir. Bu sırada Stochastic osilatör de aşağı dönmüş olsun ve fiyat serisi ile bir ayrışma göstersin. Bu durum, Stochastic modülü içinde uygulanan piyasa modellerinin birine karşılık gelmektedir. Söz konusu model, fiyatların düşüşünü öngörür. Anlamlılık değeri 80 'e eşittir.

Şimdi nihai "oylamanın" sonucunu hesaplayabiliriz. MA modülünden elde edilen oran, 0.4 * 10 = 4 şeklinde hesaplanır. Stochastic modülünden elde edilen oran, 0.8 * (-80) = -64 şeklinde hesaplanır. Son değer, bu iki oranın aritmetik ortalamasıyla bulunur: (4 - 64)/2 = -30. Oylamanın sonucu, izafi gücü 30 'a eşit olan olan satış sinyalidir. 20 değerine sahip işlem eşiği geçilmiştir. Sonuç olarak bir kısa pozisyon açılacaktır.

a) Fiyat serisi ve Stochastic osilatör ayrışması (çeşit 1 ve 2).

b) Fiyat serisinin MA göstergesini yukarı-yönlü kesmesi (çeşit 1).

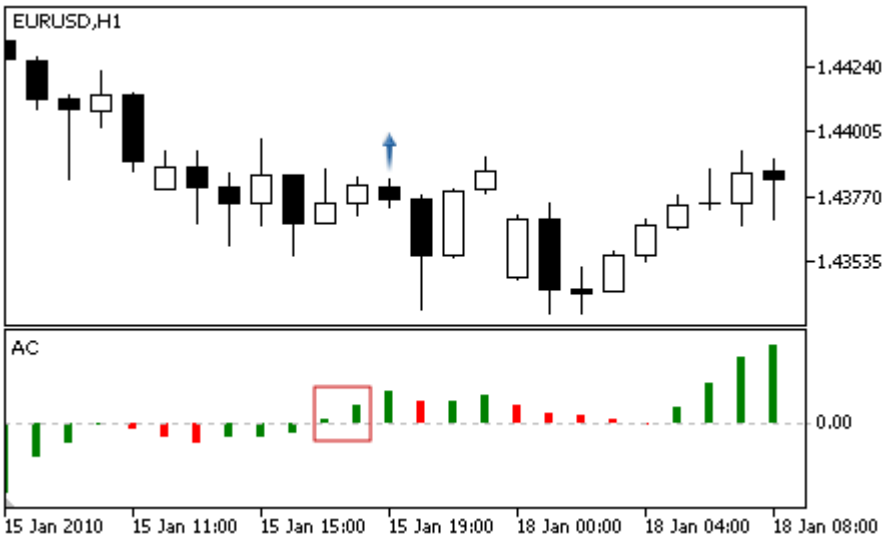
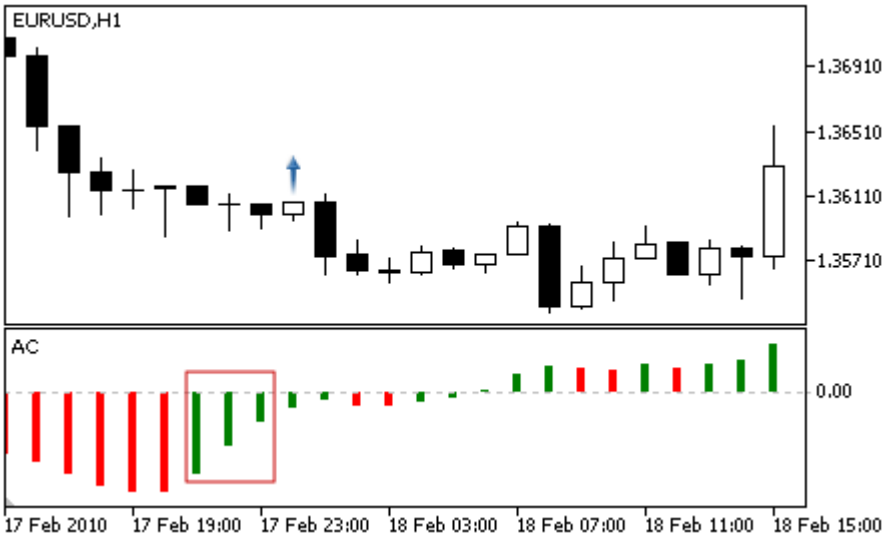
c) Fiyat serisinin MA göstergesini aşağı-yönlü kesmesi (çeşit 2).

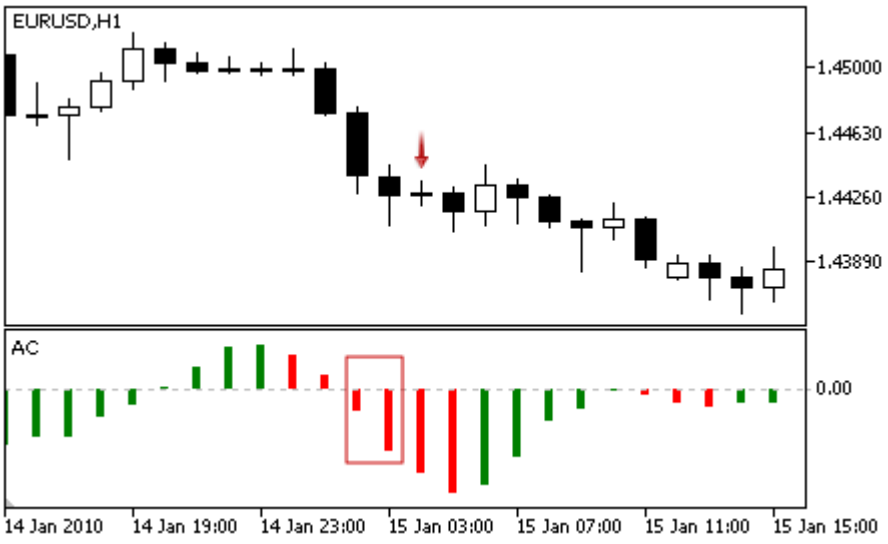
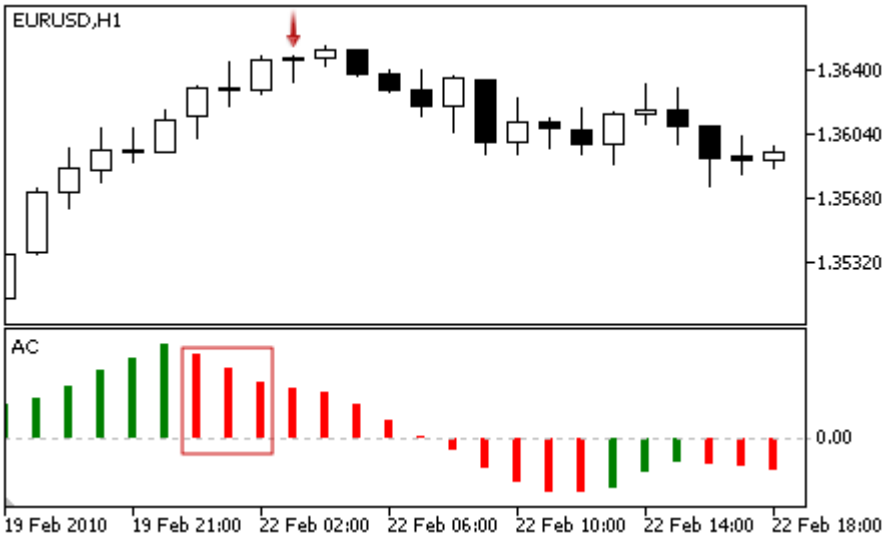
Accelerator Oscillator Göstergesinin Sinyalleri

Bu modül, Accelerator Oscillator göstergesinin piyasa modellerini temel alır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	• Gösterge değeri 0 'dan büyük, ayrıca mevcut ve önceki çubuklarda yükseliş gözleniyor.  • Gösterge değeri 0 'dan büyük, ayrıca mevcut ve önceki çubuklarda yükseliş gözleniyor. 
Satış için	• Gösterge değeri 0 'dan küçük, ayrıca mevcut ve önceki çubuklarda düşüş gözleniyor.

Sinyal Tipi	Koşulların Açıklaması
	 • Gösterge değeri 0 'dan küçük, ayrıca mevcut ve önceki çubuklarda düşüş gözleniyor. 
Alışa itiraz yok	Gösterge değeri incelenen çubuk üzerinde artış gösteriyor.
Satışa itiraz yok	Gösterge değeri incelenen çubuk üzerinde düşüş gösteriyor.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

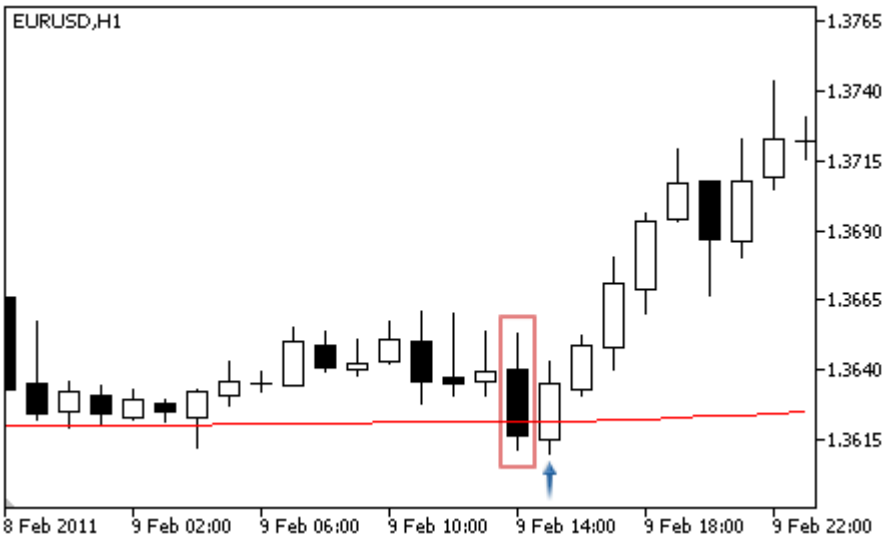
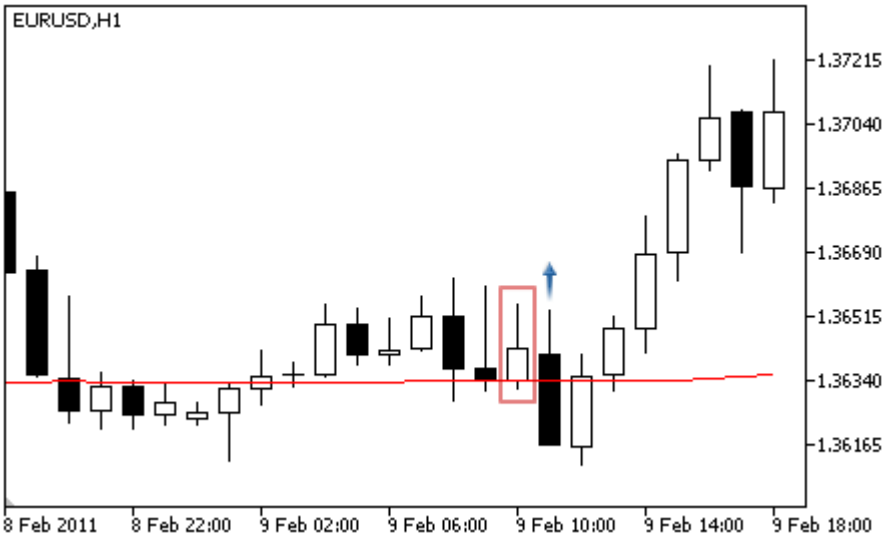
Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.

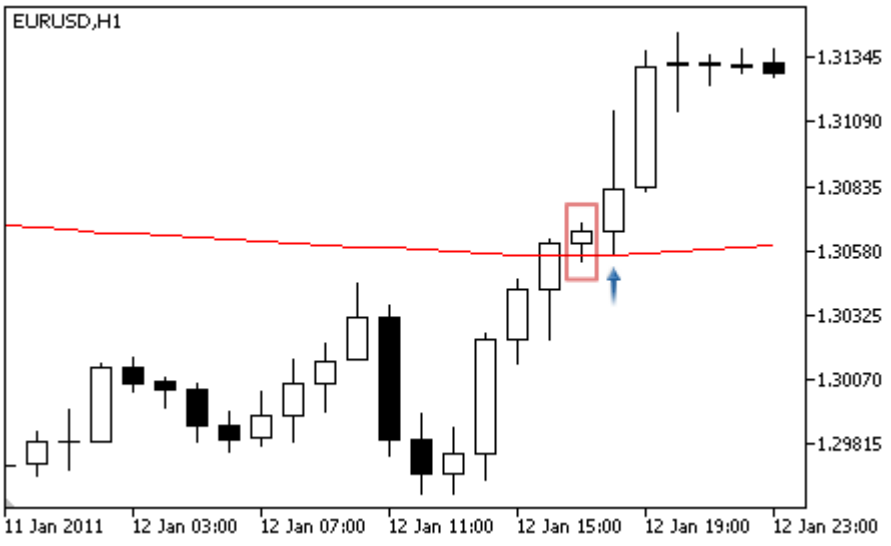
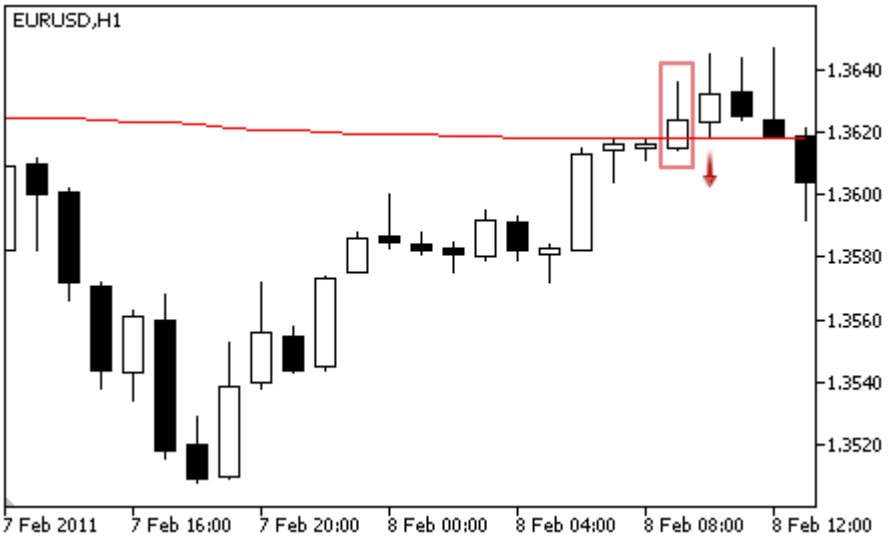
Adaptive Moving Average Göstergesinin Sinyalleri

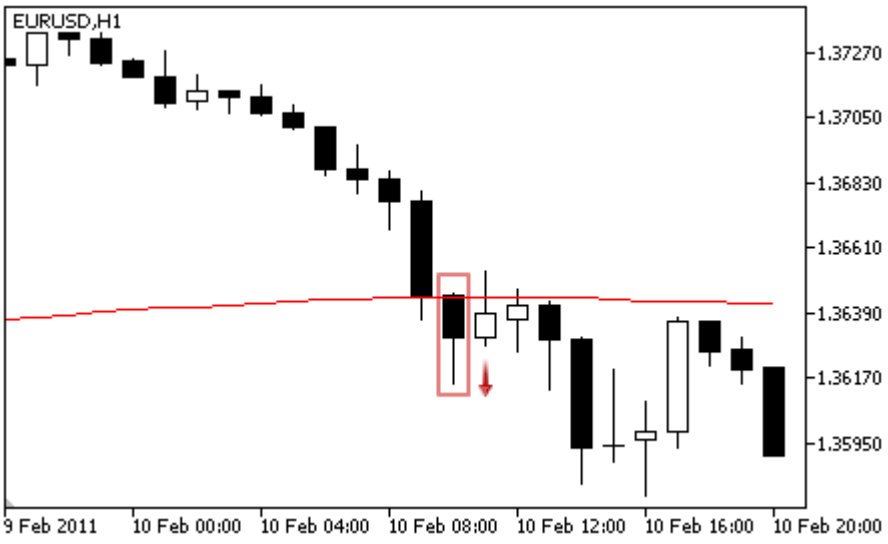
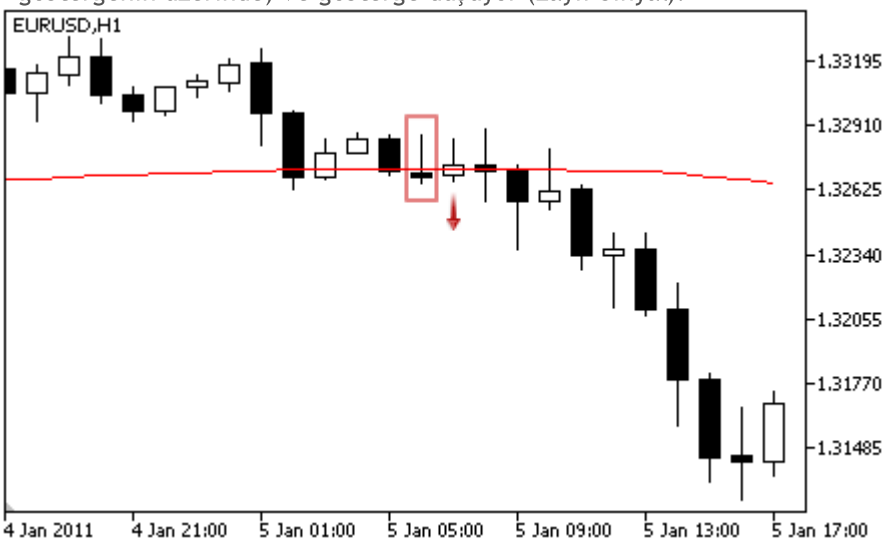
Bu modül, Adaptive Moving Average göstergesinin piyasa modellerini temel alır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	• Fiyat serisi göstergeyi aşağı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin üzerinde ve kapanış fiyatı (Close) göstergenin altında) ve gösterge yükseliyor (zayıf sinyal).  • **Hareketli Ortalama kesişimi.** Fiyat serisi göstergeyi yükarı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin altında ve kapanış fiyatı (Close) göstergenin üzerinde) ve gösterge yükseliyor (güçlü sinyal). 

Sinyal Tipi	Koşulların Açıklaması
	• Çubuğun alt gölgesi göstergeyi kesmiş (mevcut çubuğun açılış (Open) ve kapanış (Close) fiyatları göstergenin üzerinde, ve düşük (Low) fiyat göstergenin altında) ve gösterge yükseliyor (zayıf sinyal). 
Satış için	• Fiyat serisi göstergeyi yukarı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin altında ve kapanış fiyatı (Close) göstergenin üzerinde) ve gösterge değeri düşüyor (zayıf sinyal).  • **Hareketli Ortalama kesişimi.** Fiyat serisi göstergeyi aşağı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin üstünde ve kapanış fiyatı (Close) göstergenin altında) ve gösterge değeri düşüyor (güçlü sinyal).

Sinyal Tipi	Koşulların Açıklaması
	 • Çubuğun üst gölgesi göstergeyi kesmiş (mevcut çubuğun açılış (Open) ve kapanış (Close) fiyatları göstergenin altında, ve yüksek (High) fiyat göstergenin üzerinde) ve gösterge düşüyor (zayıf sinyal). 
Alışa itiraz yok	Fiyat göstergenin üzerinde.
Satışa itiraz yok	Fiyat göstergenin altında.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

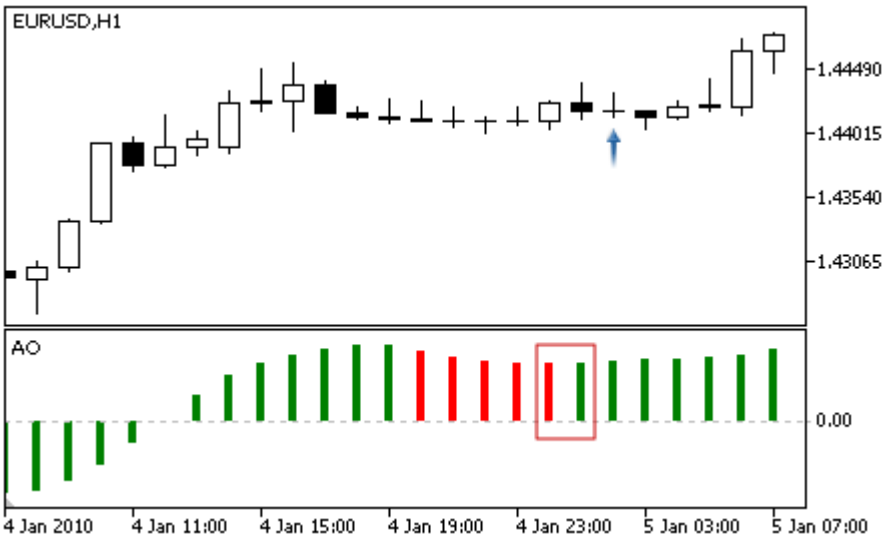
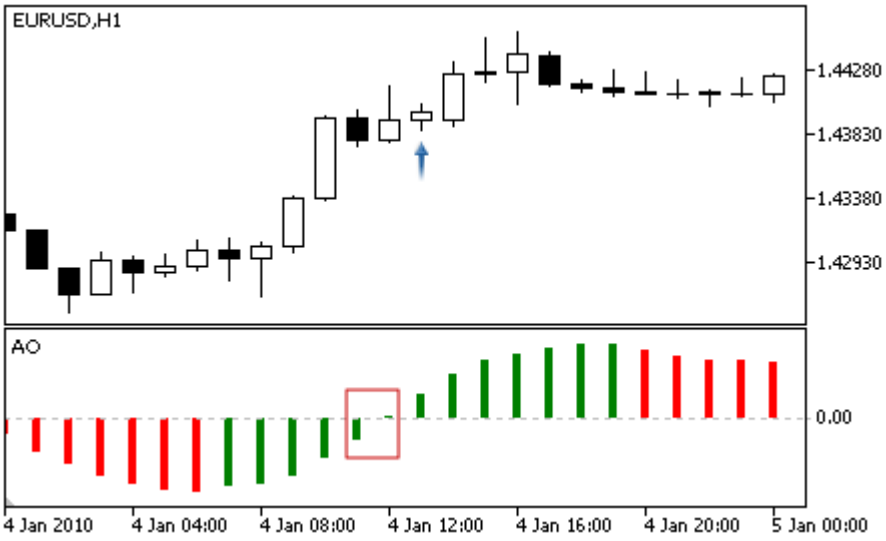
Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.
PeriodMA	Göstergenin hareketli ortalama periyodu.
Shift	Göstergenin zaman ekseni üzerindeki kaydırma değeri (çubuk bazında).
Method	Ortalama yöntemi.
Applied	Göstergenin hesaplanmasında kullanılan fiyat serisi.

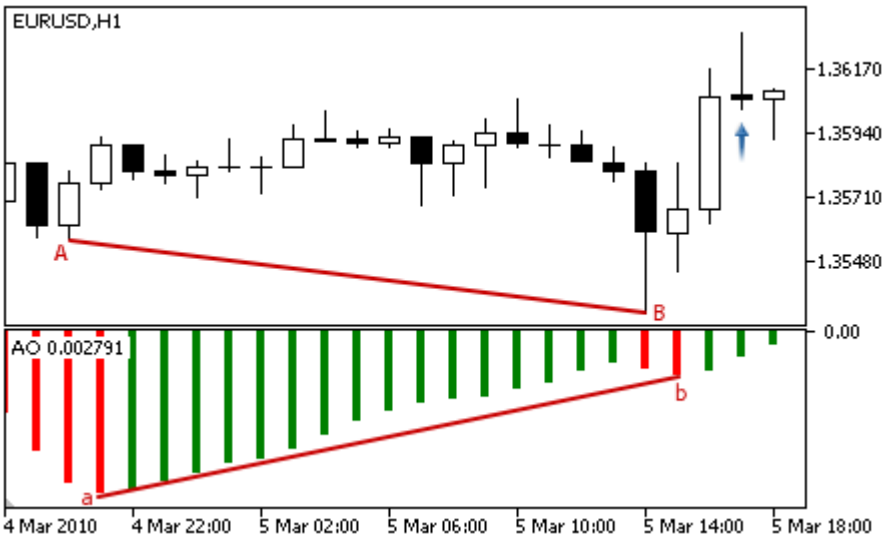
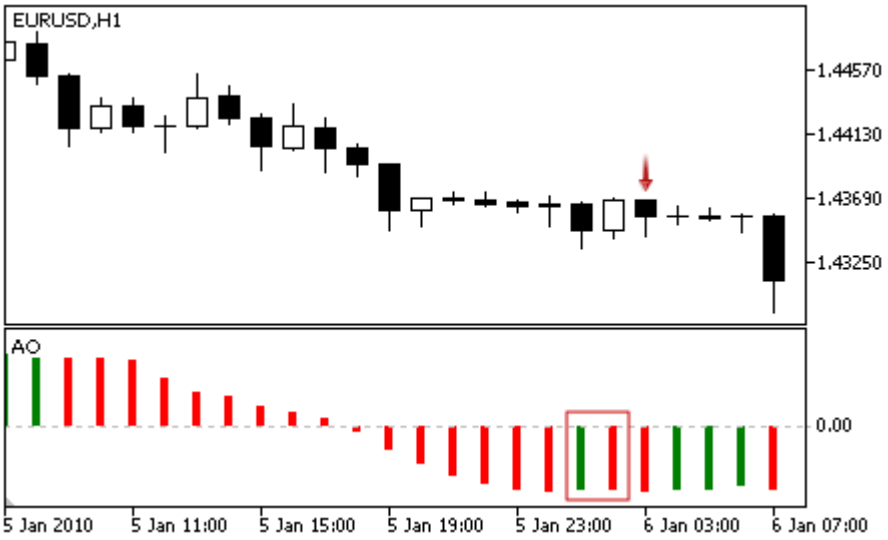
Awesome Oscillator göstergesinin sinyalleri

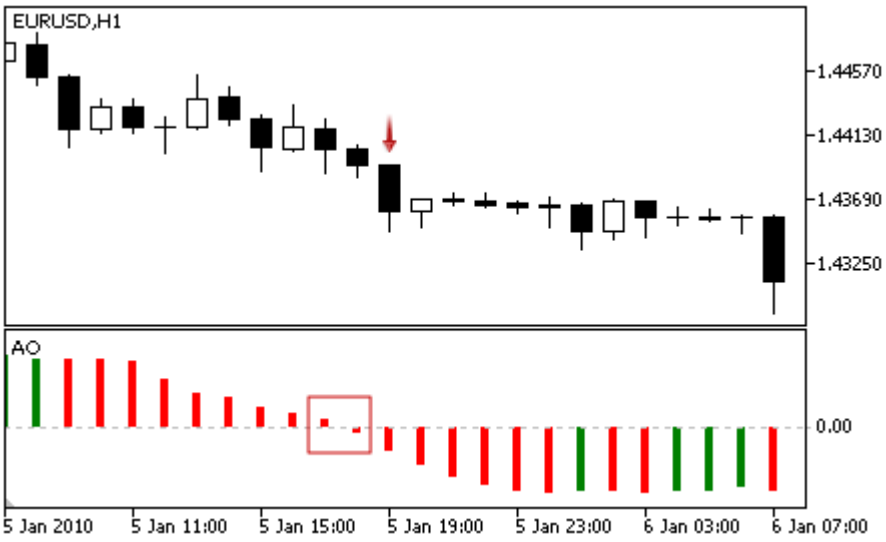
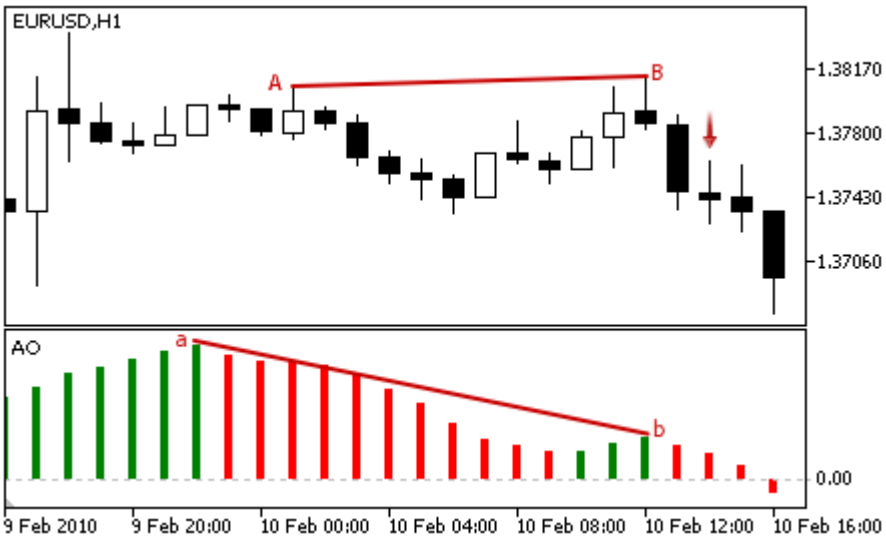
Bu sinyal modülü Awesome Oscillator göstergesinin piyasa modelini temel alır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	• **Çay-Tabağı** — önceki çubukta göstergenin değeri düşmüş ve mevcut çubukta yükseliyor, her iki çubuk için de gösterge değeri 0 'dan büyük.  • **Sıfır kesimi** — göstergenin değeri mevcut çubuk için 0 'dan büyük ve bir önceki çubuk için sıfırdan küçük.  • **Ayrışma** — göstergenin analiz edilen ilk tabanı bir öncekinden daha yüksek, ve karşılık gelen fiyat tabanı bir öncekinden daha düşük. Ayrıca, gösterge sıfır seviyesinin altında olmalı.

Sinyal Tipi	Koşulların Açıklaması
	
Satış için	• **Çay-Tabağı** — önceki çubukta göstergenin değeri yükselmiş ve mevcut çubukta düşüyor, her iki çubuk için de gösterge değeri 0 'dan küçük.  • **Sıfır kesimi** — göstergenin değeri mevcut çubuk için 0 'dan küçük ve bir önceki çubuk için sıfırdan büyük.

Sinyal Tipi	Koşulların Açıklaması
	 • **Ayrışma** — göstergenin analiz edilen ilk zirvesi bir öncekinden daha düşük, ve karşılık gelen fiyat zirvesi bir öncekinden daha yüksek. Ayrıca, gösterge sıfır seviyesinin üstünde olmalı. 
Alışa itiraz yok	Gösterge değeri incelenen çubuk üzerinde artış gösteriyor.
Satışa itiraz yok	Gösterge değeri incelenen çubuk üzerinde düşüş gösteriyor.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

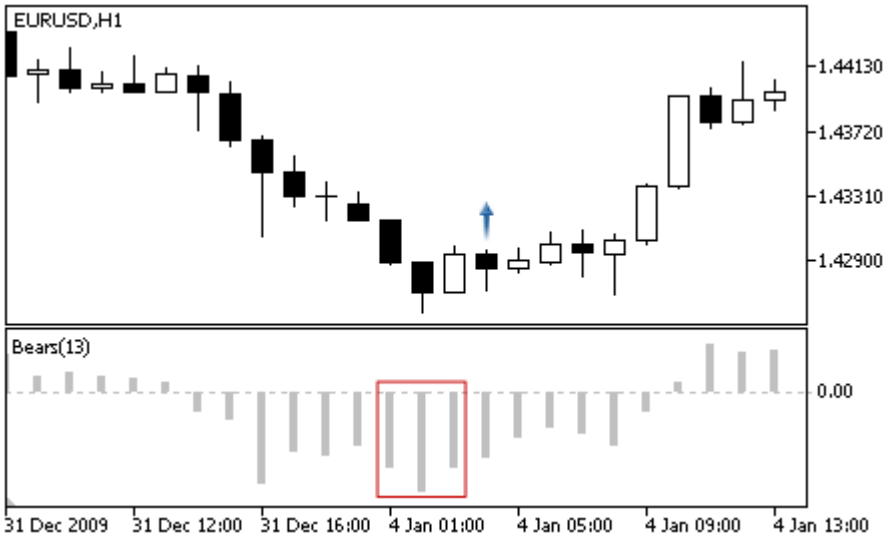
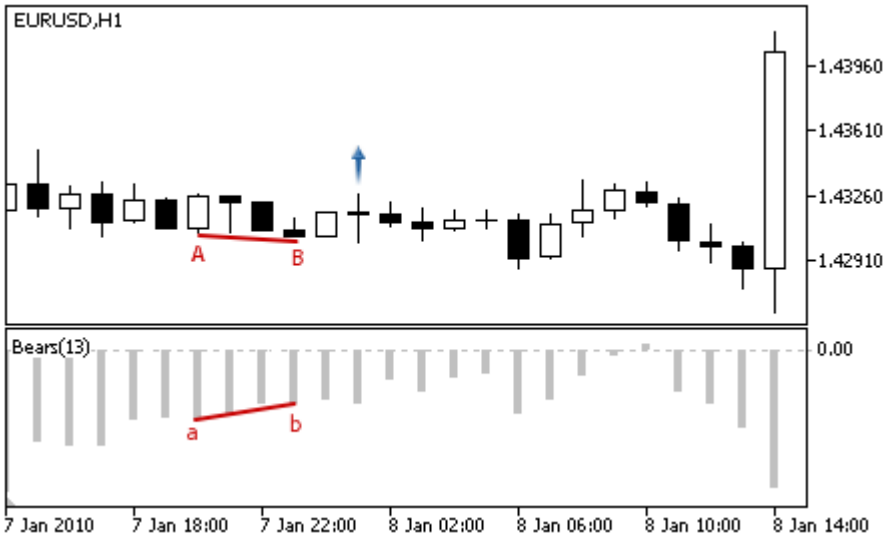
Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.

Bears Power Osilatörünün Sinyalleri

Bu sinyal modülü Bears Power osilatörünün piyasa modellerine dayanır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	• **Ters** — Osilatör yukarı doğru yönelmiş ve incelenen çubuk üzerindeki değeri 0 'dan küçük.  • **Ayrışma** — osilatörün analiz edilen ilk tabanı bir öncekinden daha yüksek, ve karşılık gelen fiyat tabanı bir öncekinden daha düşük. Ayrıca, Osilatör değeri sıfır seviyesinin altında olmalı. 
Satış için	Satış sinyali yok.
Alışa itiraz yok	Osilatör değeri 0 'dan küçük.

Sinyal Tipi	Koşulların Açıklaması
Satışa itiraz yok	Sinyal yok.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

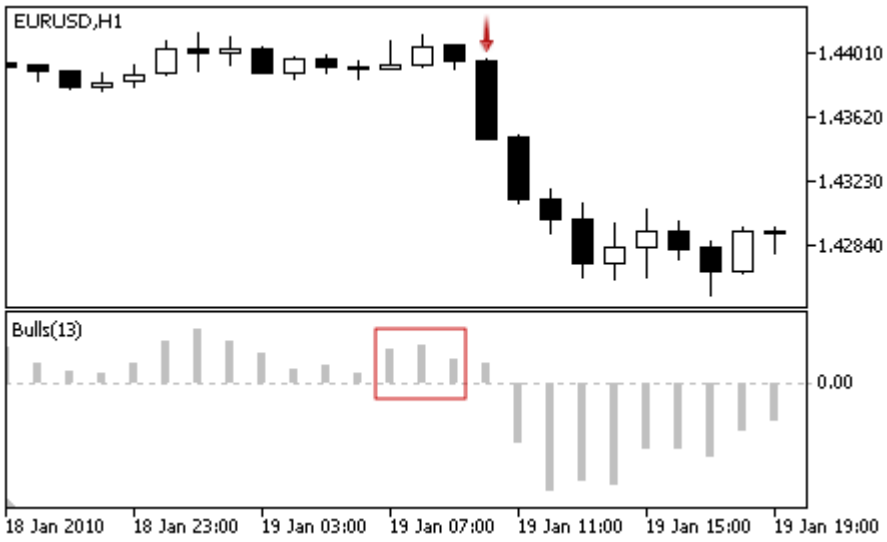
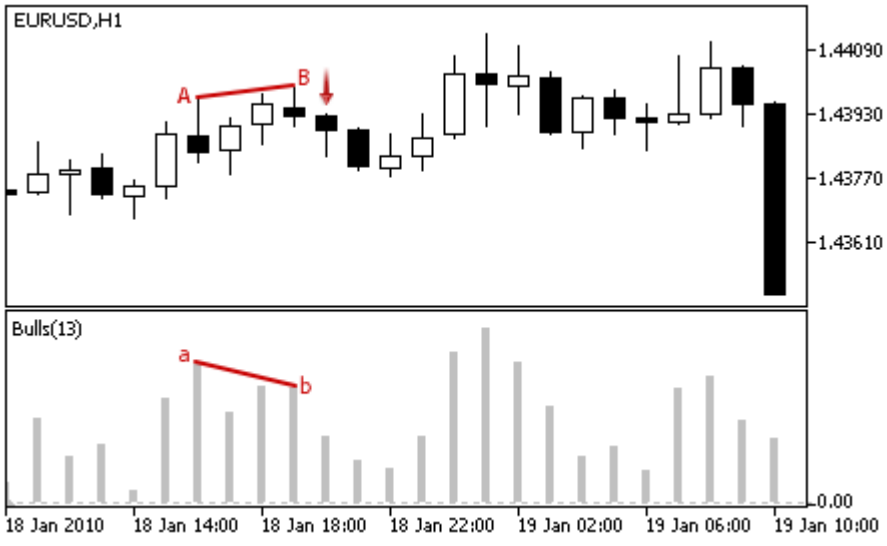
Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.
PeriodBears	Osilatörün hesaplanmasında kullanılan periyot değeri.

ulls Power Osilatörünün Sinyalleri

Bu sinyal modülü Bulls Power osilatörünün piyasa modellerine dayanır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	Alış sinyali yok.
Satış için	• **Ters** — Osilatör aşağı doğru yönelmiş ve incelenen çubuk üzerindeki değeri 0 'dan büyük.  • **Ayrışma** — osilatörün analiz edilen ilk zirvesi bir öncekinden daha düşük, ve karşılık gelen fiyat zirvesi bir öncekinden daha yüksek. Ayrıca, osilatör sıfır seviyesinin üstünde olmalı. 
Alışa itiraz yok	Sinyal yok.

Sinyal Tipi	Koşulların Açıklaması
Satışa itiraz yok	Osilatör değeri 0 'dan büyük.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

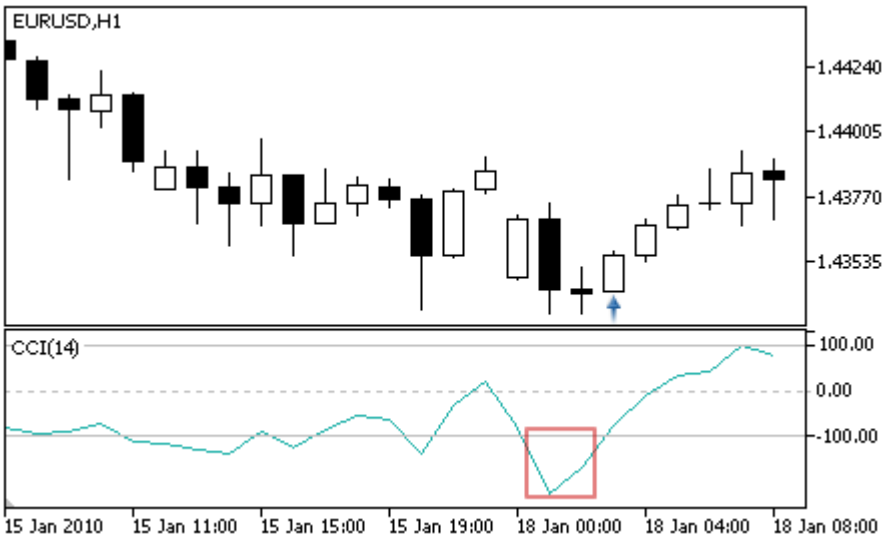
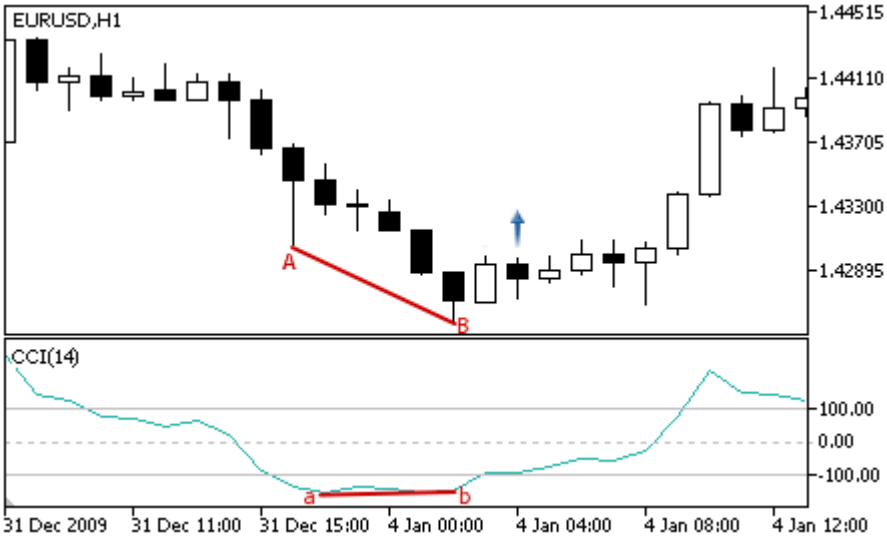
Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.
PeriodBulls	Osilatörün hesaplanmasında kullanılan periyot değeri.

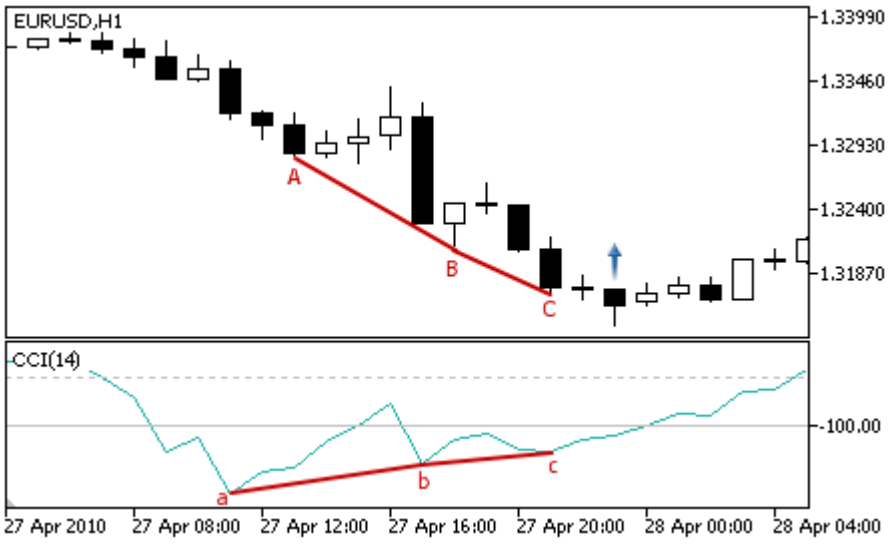
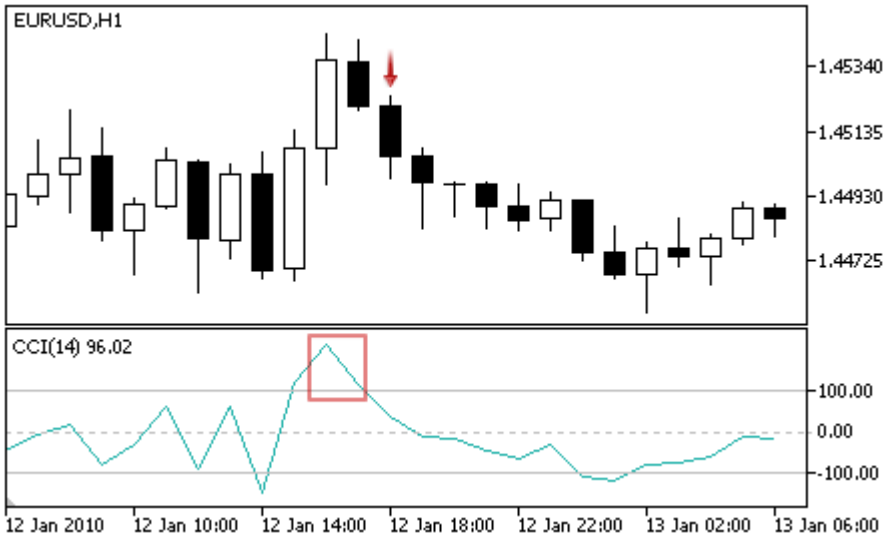
Commodity Channel Index Osilatörünün Sinyalleri

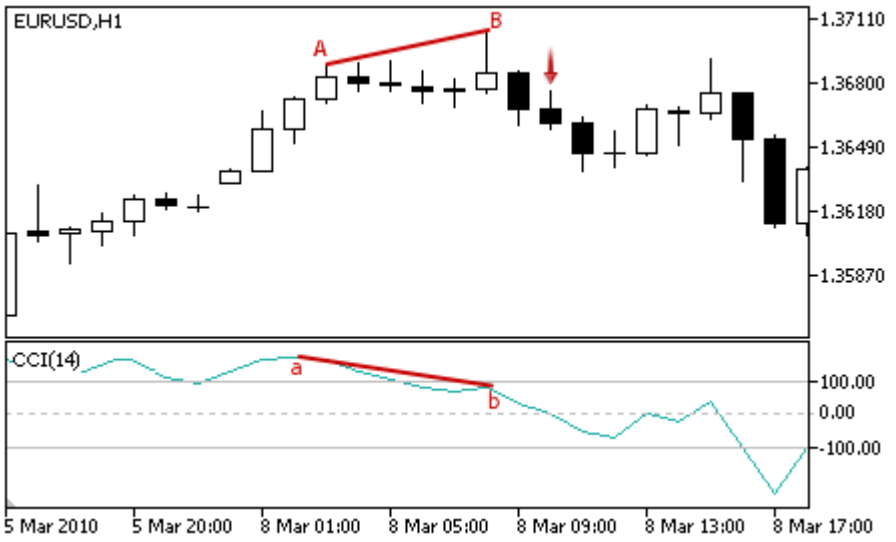
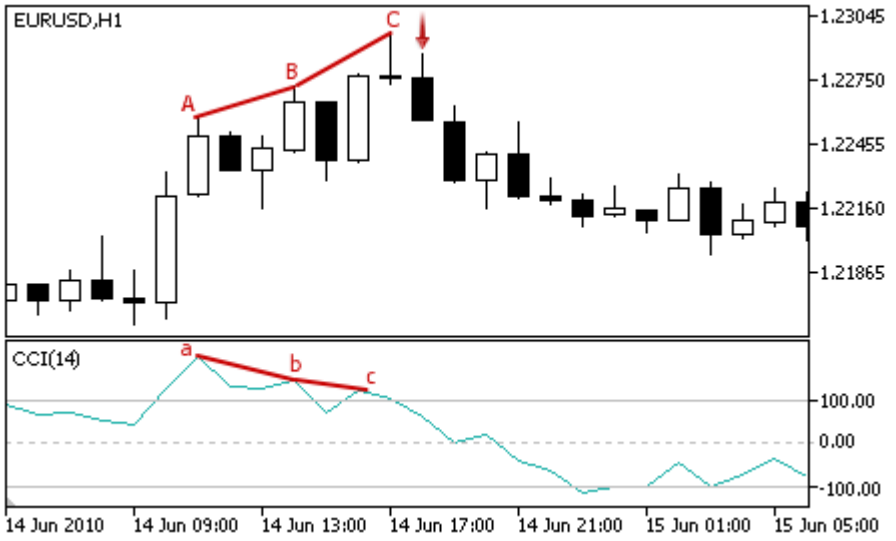
Bu sinyal modülü, Commodity Channel Index osilatörünün piyasa modellerine dayanır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	• **Aşırı-satış seviyesinden dönüş** — osilatör yukarı yönelmiş ve incelenen mevcut çubuk üzerindeki değeri aşırı-satış seviyesini geçmiş (varsayılan değer : 100).  • **Ayrışma** — osilatörün analiz edilen ilk tabanı bir öncekinden daha yüksek, ve karşılık gelen fiyat tabanı bir öncekinden daha düşük.  • **Çift ayrışma** — osilatörde üç ardışık taban şekillenmiş, her biri bir öncekinden daha yüksek; aynı şekilde fiyat serisi de üç ardışık tabana sahip ve her biri bir öncekinden daha derin.

Sinyal Tipi	Koşulların Açıklaması
	
Satış için	• **Aşırı-alış seviyesinden dönüş** — osilatör aşağı yönelmiş ve incelenen mevcut çubuk üzerindeki değeri aşırı-alış seviyesini geçmiş (varsayılan değer : 100).  • **Ayrışma** — osilatörün analiz edilen ilk zirvesi bir öncekinden daha düşük, ve karşılık gelen fiyat zirvesi bir öncekinden daha yüksek.

Sinyal Tipi	Koşulların Açıklaması
	 • **Çift ayrışma** — osilatörde üç ardışık zirve şekillenmiş, her biri bir öncekinden daha düşük; aynı şekilde fiyat serisi de üç ardışık zirveye sahip ve her biri bir öncekinden daha yüksek. 
Alışa itiraz yok	Osilatör değeri incelenen çubuk üzerinde artıyor.
Satışa itiraz yok	Osilatör değeri incelenen çubuk üzerinde düşüyor.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

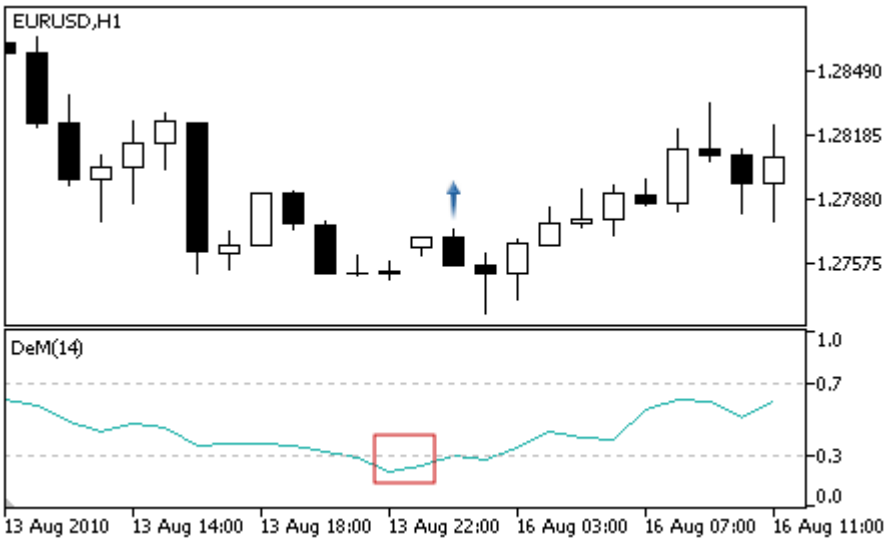
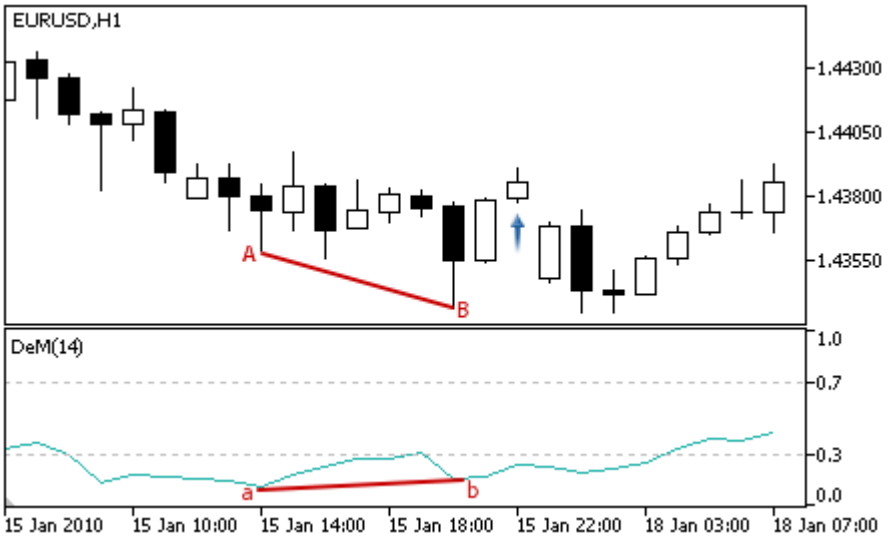
Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.
PeriodCCI	Osilatörün hesaplanmasında kullanılan periyot değeri.
Applied	Osilatörün hesaplanmasında kullanılan fiyat serileri.

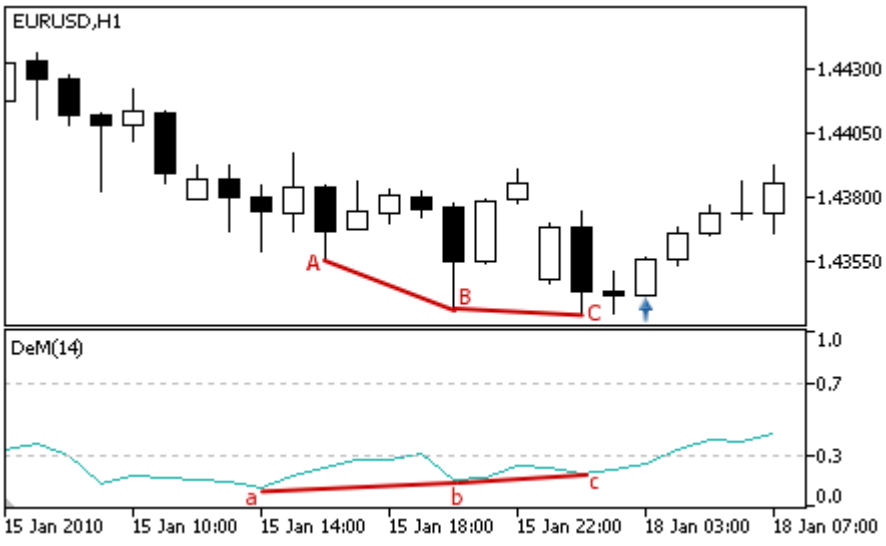
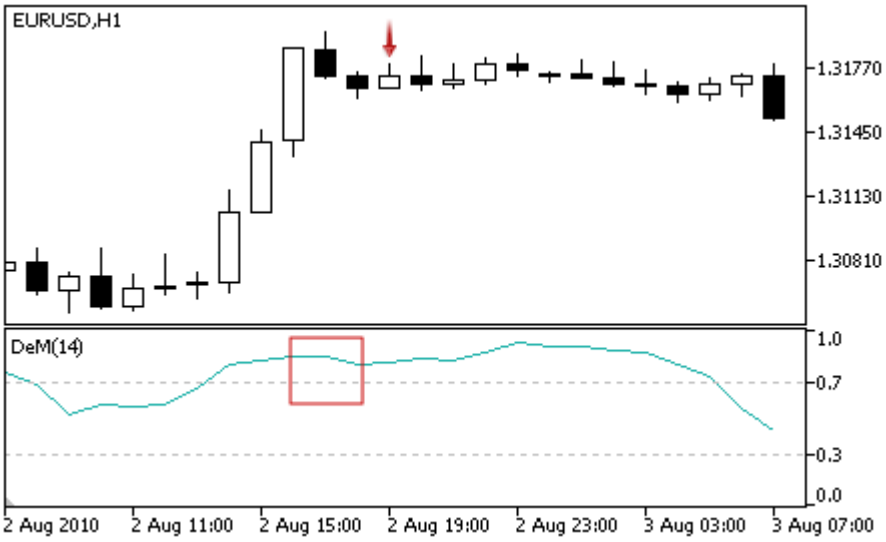
DeMarker Osilatörünün Sinyalleri

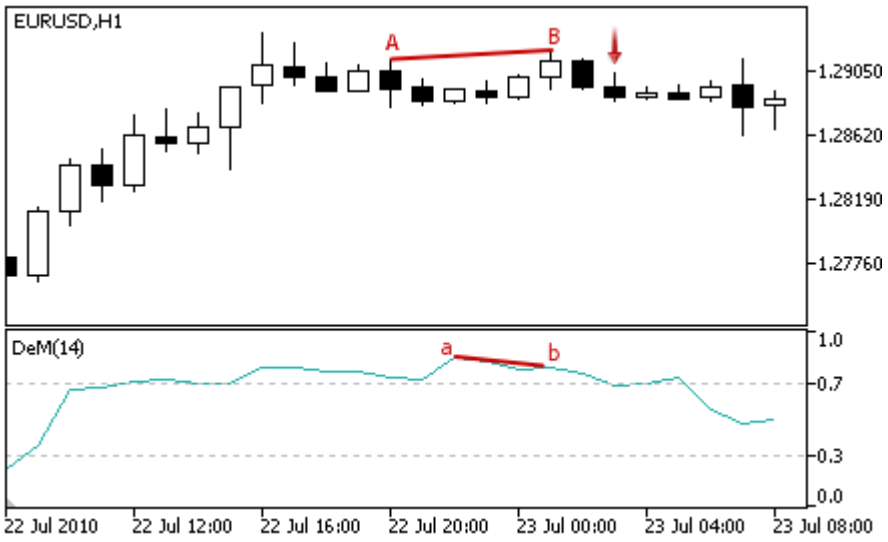
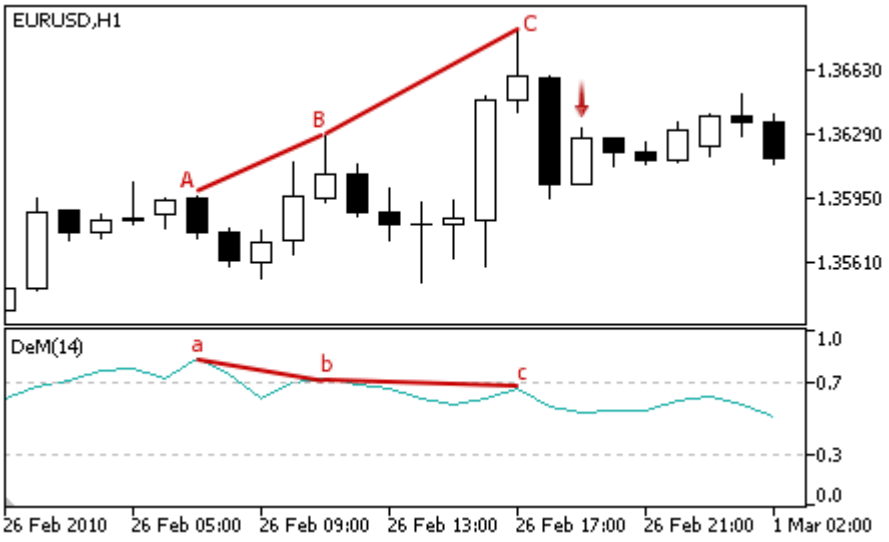
Bu sinyal modülü, DeMarker osilatörünün piyasa modellerini temel alır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	• **Aşırı-satış seviyesinin ardından ters dönüş** — osilatör yukarı yönelmiş ve incelenen mevcut çubuk üzerindeki değeri aşırı-satış seviyesini geçmiş (varsayılan değer : 0.3).  • **Ayrışma** — osilatörün analiz edilen ilk tabanı bir öncekinden daha yüksek, ve karşılık gelen fiyat tabanı bir öncekinden daha düşük.  • **Çift ayrışma** — osilatörde üç ardışık taban şekillenmiş, her biri bir öncekinden daha yüksek; aynı şekilde fiyat serisi de üç ardışık tabana sahip ve her biri bir öncekinden daha derin.

Sinyal Tipi	Koşulların Açıklaması
	
Satış için	• **Aşırı-alım seviyesinin ardından ters dönüş** — osilatör aşağı yönelmiş ve incelenen mevcut çubuk üzerindeki değeri aşırı-alış seviyesini geçmiş (varsayılan değer : 0.7).  • **Ayrışma** — osilatörün analiz edilen ilk zirvesi bir öncekinden daha düşük, ve karşılık gelen fiyat zirvesi bir öncekinden daha yüksek.

Sinyal Tipi	Koşulların Açıklaması
	 • **Çift ayrışma** — osilatörde üç ardışık zirve şekillenmiş, her biri bir öncekinden daha düşük; aynı şekilde fiyat serisi de üç ardışık zirveye sahip ve her biri bir öncekinden daha yüksek. 
Alışa itiraz yok	Osilatör değeri incelenen çubuk üzerinde artıyor.
Satışa itiraz yok	Osilatör değeri incelenen çubuk üzerinde düşüyor.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

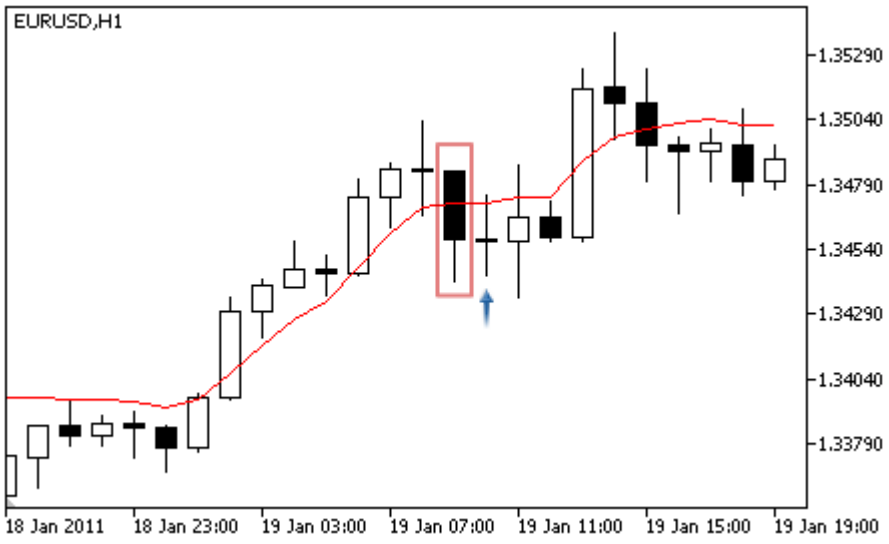
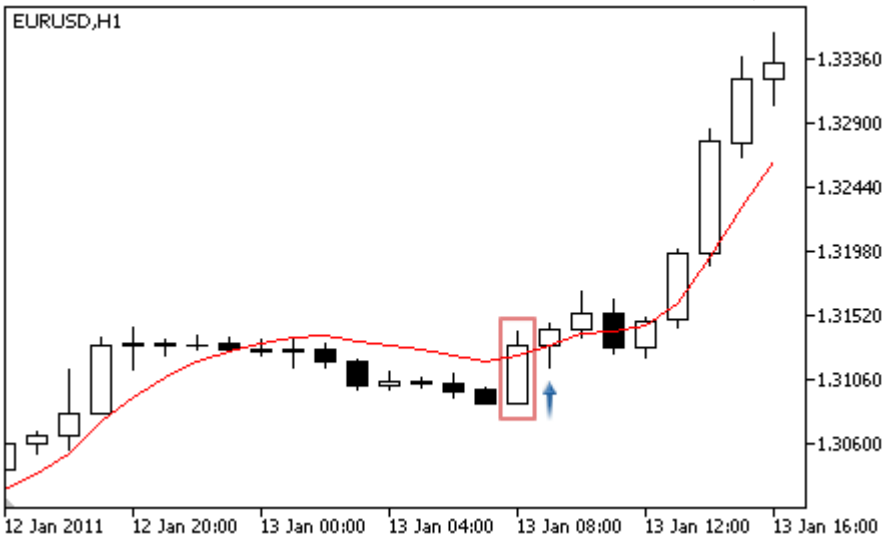
Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.
PeriodDeM	Osilatörün hesaplanmasında kullanılan periyot değeri.

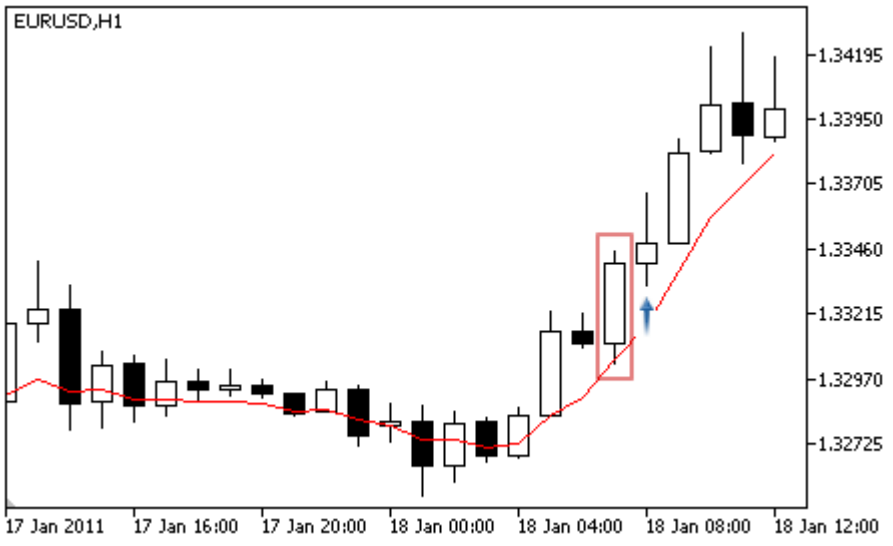
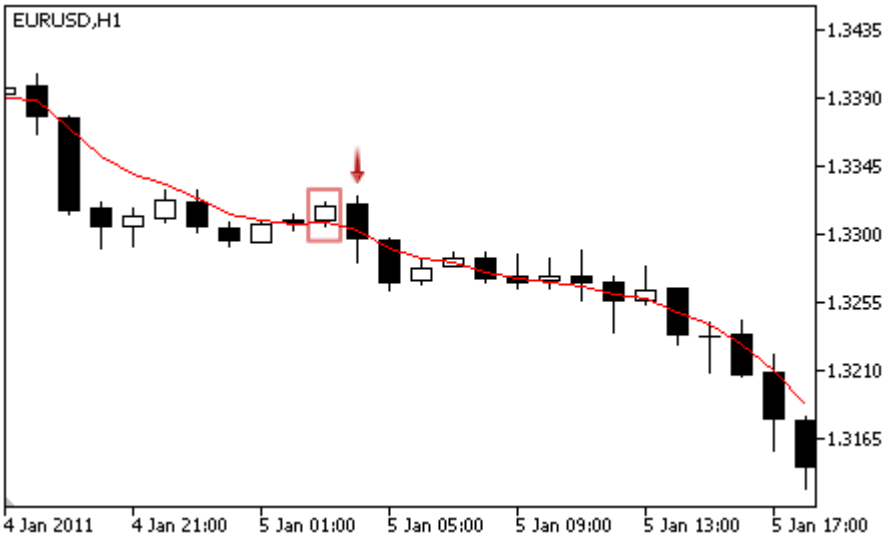
Double Exponential Moving Average Göstergesinin Sinyalleri

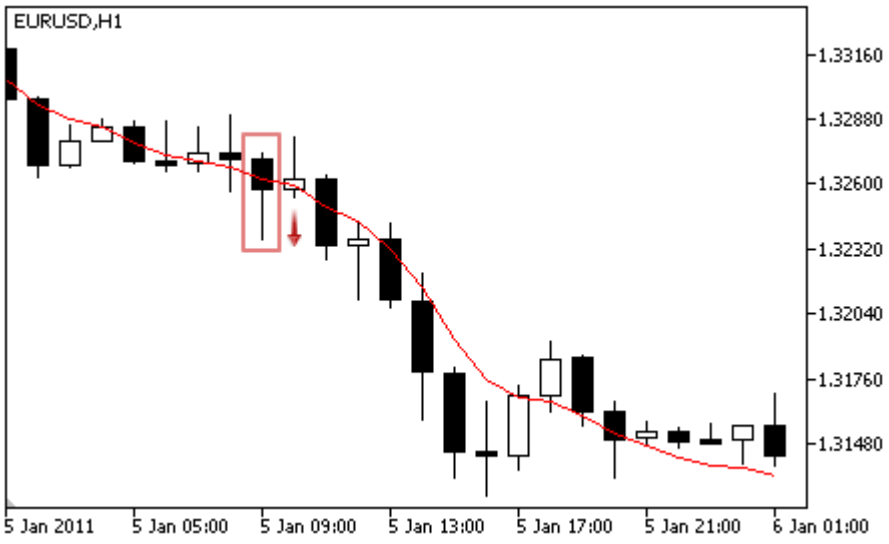
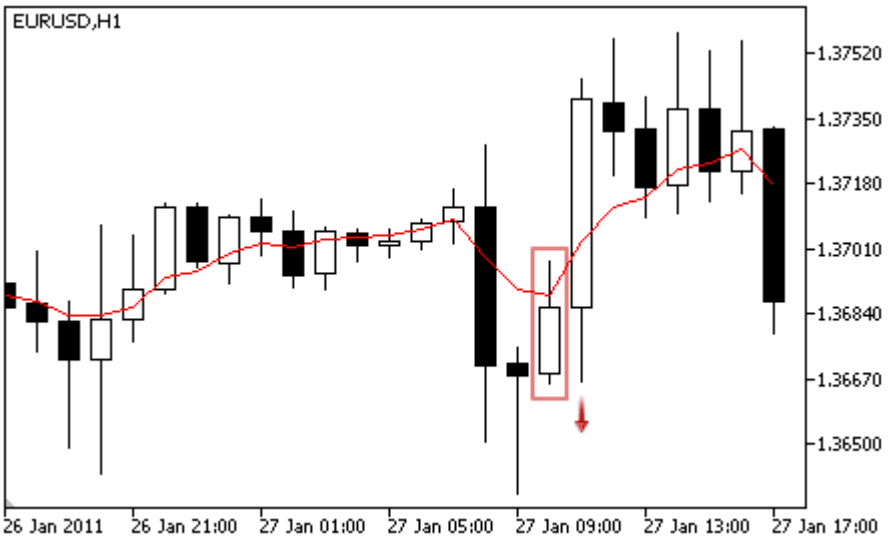
Bu modül, Double Exponential Moving Average göstergesinin piyasa modellerini temel alır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	• Fiyat serisi göstergeyi aşağı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin üzerinde ve kapanış fiyatı (Close) göstergenin altında) ve gösterge yükseliyor (zayıf sinyal).  • **Hareketli Ortalama kesişimi.** Fiyat serisi göstergeyi yukarı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin altında ve kapanış fiyatı (Close) göstergenin üzerinde) ve gösterge yükseliyor (güçlü sinyal). 

Sinyal Tipi	Koşulların Açıklaması
	• Çubuğun alt gölgesi göstergeyi kesmiş (mevcut çubuğun açılış (Open) ve kapanış (Close) fiyatları göstergenin üzerinde, ve düşük (Low) fiyat göstergenin altında) ve gösterge yükseliyor (zayıf sinyal). 
Satış için	• Fiyat serisi göstergeyi yukarı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin altında ve kapanış fiyatı (Close) göstergenin üzerinde) ve gösterge değeri düşüyor (zayıf sinyal).  • **Hareketli Ortalama kesişimi.** Fiyat serisi göstergeyi aşağı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin üstünde ve kapanış fiyatı (Close) göstergenin altında) ve gösterge değeri düşüyor (güçlü sinyal).

Sinyal Tipi	Koşulların Açıklaması
	 • Çubuğun üst gölgesi göstergeyi kesmiş (mevcut çubuğun açılış (Open) ve kapanış (Close) fiyatları göstergenin altında, ve yüksek (High) fiyat göstergenin üzerinde) ve gösterge düşüyor (zayıf sinyal). 
Alışa itiraz yok	Fiyat göstergenin yukarısındadır.
Satışa itiraz yok	Fiyat göstergenin altında.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

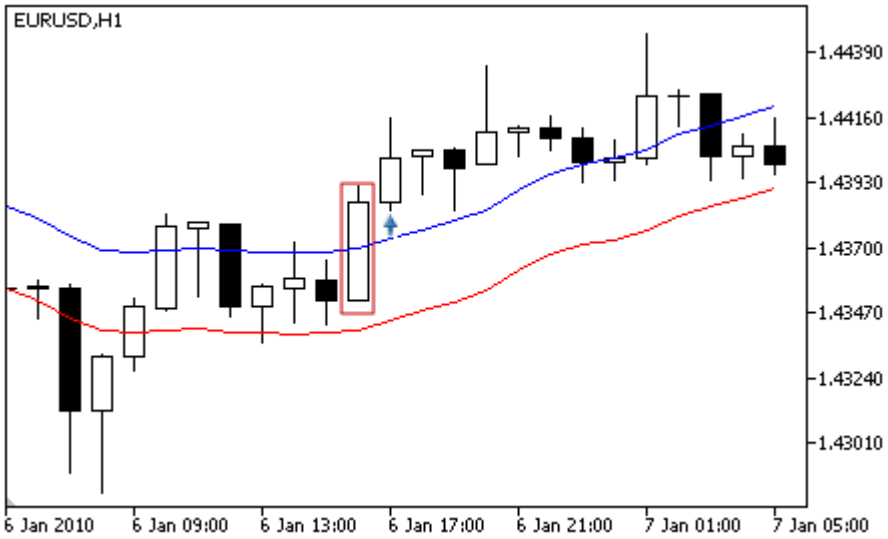
Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.
PeriodMA	Göstergenin hareketli ortalama periyodu.
Shift	Göstergenin zaman ekseni üzerindeki kaydırma değeri (çubuk bazında).
Method	Ortalama yöntemi.
Applied	Göstergenin hesaplanmasında kullanılan fiyat serisi.

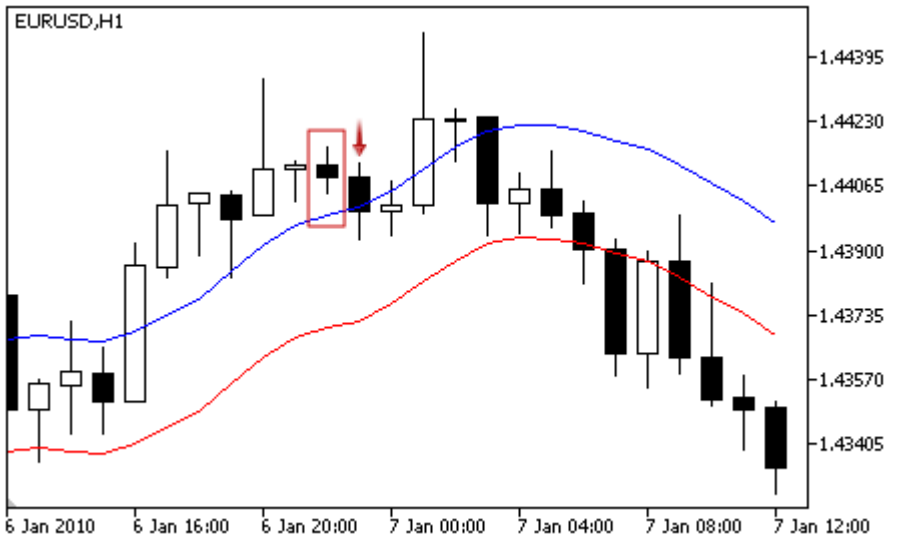
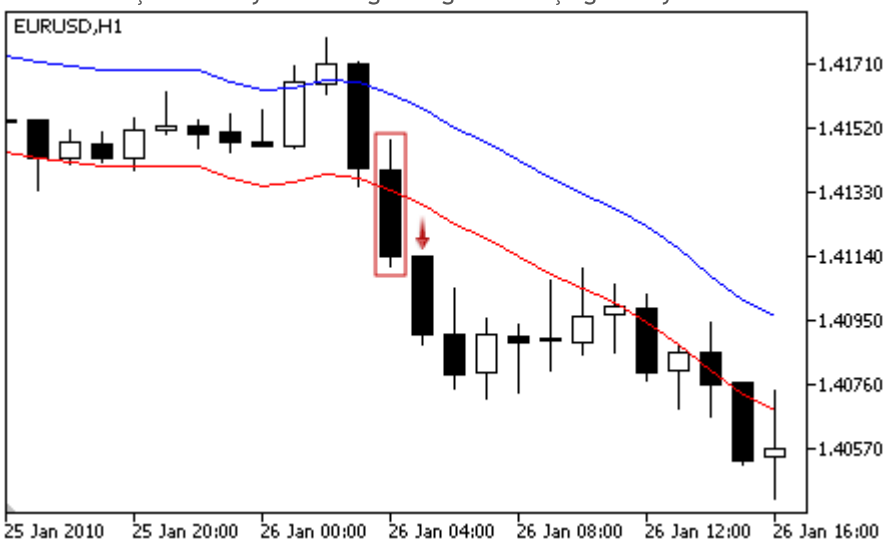
Envelopes göstergesinin Sinyalleri

Bu sinyal modül Envelopes göstergesinin piyasa modellerini temel alır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	• İncelenen çubukta fiyat serisi göstergenin alt çizgisine yakındır.  • İncelenen çubukta fiyat serisi göstergenin üst çizgisine yakındır. 
Satış için	• İncelenen çubukta fiyat serisi göstergenin üst çizgisine yakındır.

Sinyal Tipi	Koşulların Açıklaması
	 • İncelenen çubukta fiyat serisi göstergenin alt çizgisine yakındır. 
Alışa itiraz yok	Sinyal yok.
Satışa itiraz yok	Sinyal yok.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.

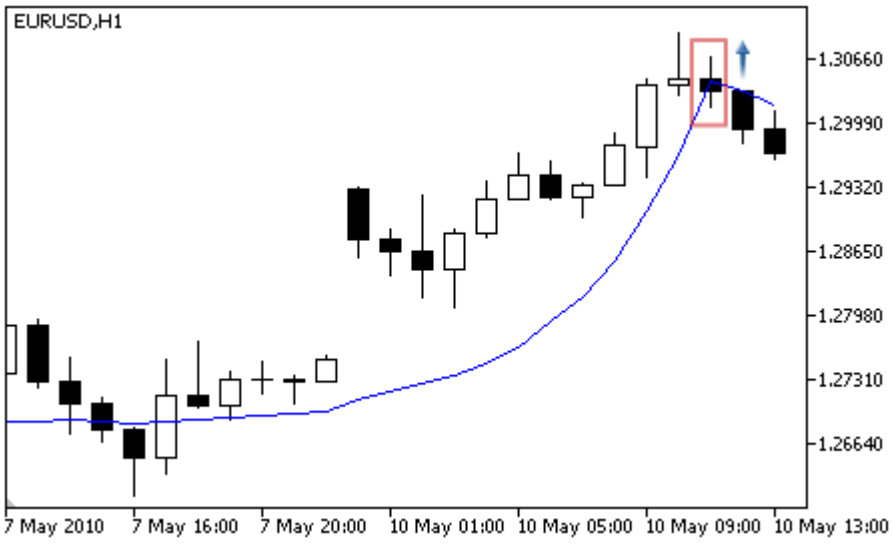
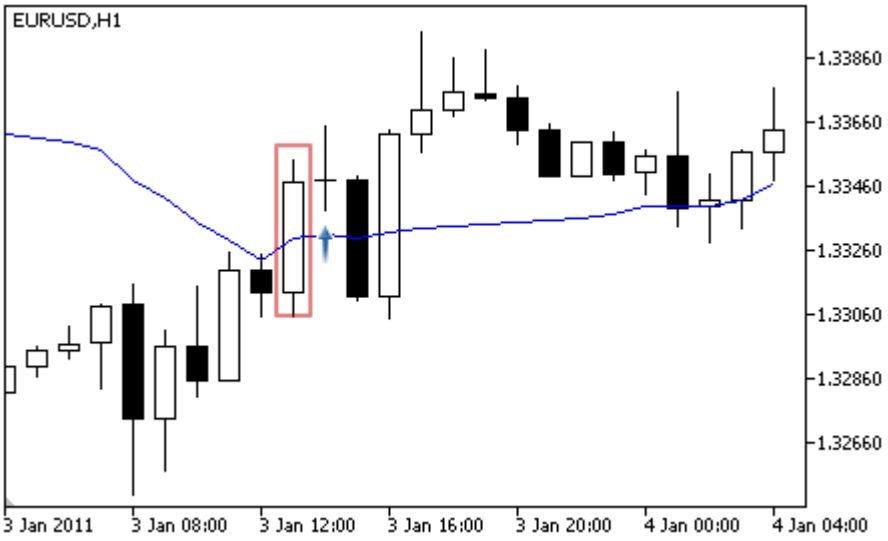
Parameter	Açıklama
PeriodMA	Göstergenin hesaplama periyodu.
Shift	Göstergenin zaman ekseni üzerindeki kaydırma değeri (çubuk bazında).
Method	Ortalama yöntemi.
Applied	Göstergenin hesaplanmasında kullanılan fiyat serisi.
Deviation	Yüzdelik cinsten, zarf kenarlarının merkez çizgiye (MA) göre sapması.

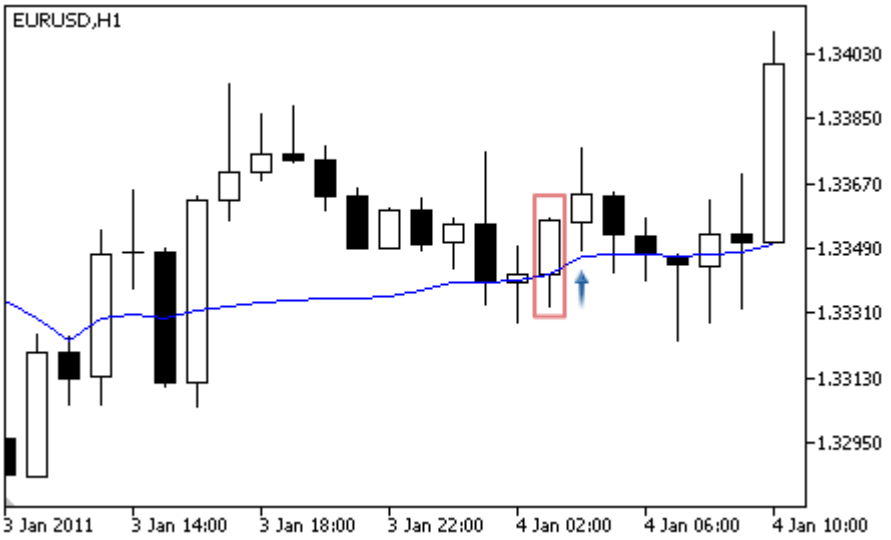
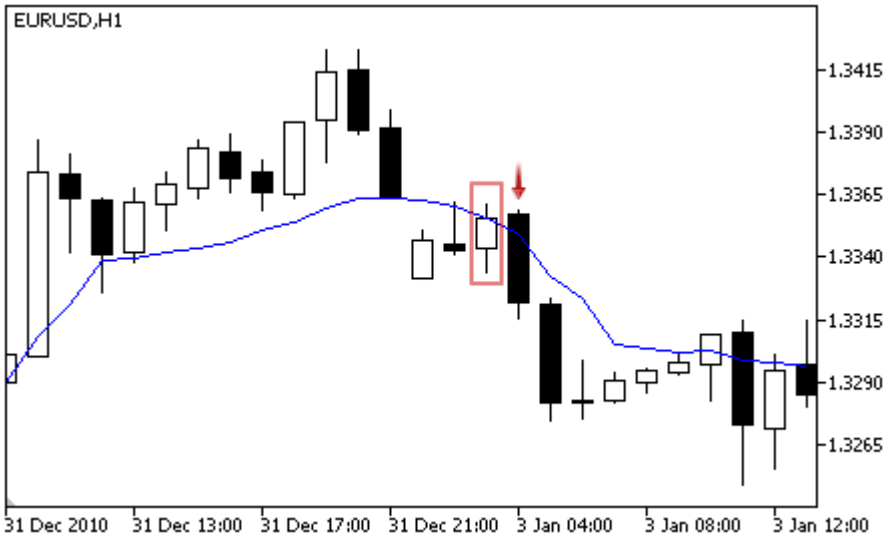
Fractal Adaptive Moving Average Göstergesinin Sinyalleri

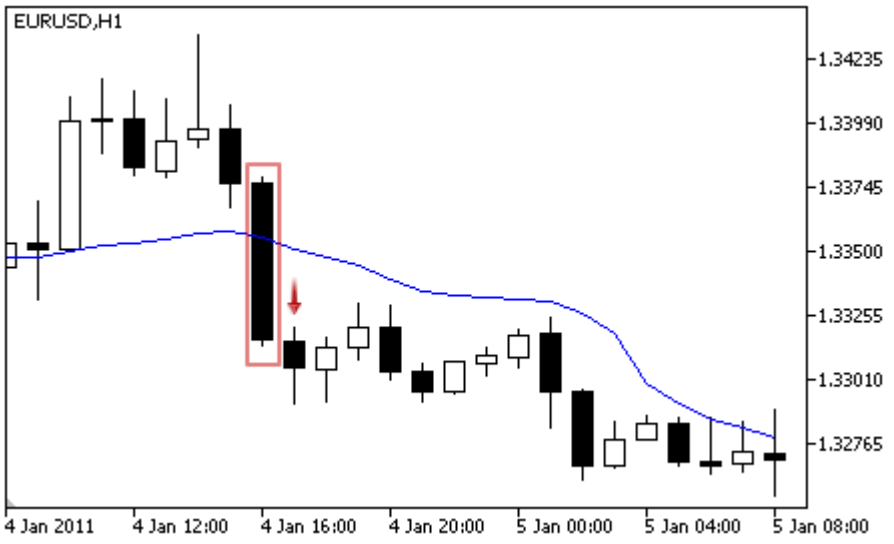
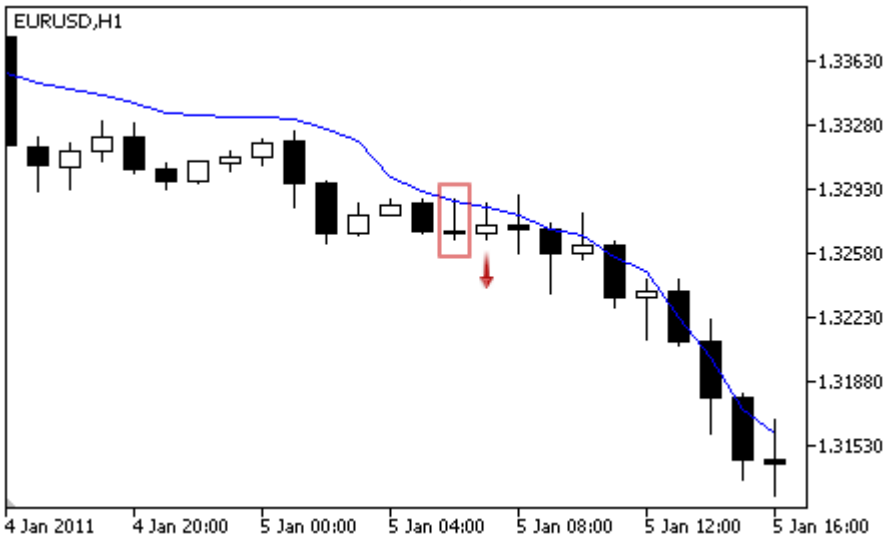
Bu sinyal modülü, Fractal Adaptive Moving Average göstergesinin piyasa modellerini temel alır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	• Fiyat serisi göstergeyi aşağı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin üzerinde ve kapanış fiyatı (Close) göstergenin altında) ve gösterge yükseliyor (zayıf sinyal).  • **Hareketli Ortalama kesişimi.** Fiyat serisi göstergeyi yükarı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin altında ve kapanış fiyatı (Close) göstergenin üzerinde) ve gösterge yükseliyor (güçlü sinyal). 

Sinyal Tipi	Koşulların Açıklaması
	• Çubuğun alt gölgesi göstergeyi kesmiş (mevcut çubuğun açılış (Open) ve kapanış (Close) fiyatları göstergenin üzerinde, ve düşük (Low) fiyat göstergenin altında) ve gösterge yükseliyor (zayıf sinyal). 
Satış için	• Fiyat serisi göstergeyi yukarı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin altında ve kapanış fiyatı (Close) göstergenin üzerinde) ve gösterge değeri düşüyor (zayıf sinyal).  • **Hareketli Ortalama kesişimi.** Fiyat serisi göstergeyi aşağı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin üstünde ve kapanış fiyatı (Close) göstergenin altında) ve gösterge değeri düşüyor (güçlü sinyal).

Sinyal Tipi	Koşulların Açıklaması
	 • Çubuğun üst gölgesi göstergeyi kesmiş (mevcut çubuğun açılış (Open) ve kapanış (Close) fiyatları göstergenin altında, ve yüksek (High) fiyat göstergenin üzerinde) ve gösterge düşüyor (zayıf sinyal). 
Alışa itiraz yok	Fiyat göstergenin üzerinde.
Satışa itiraz yok	Fiyat göstergenin altında.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.
PeriodMA	Göstergenin hareketli ortalama periyodu.
Shift	Göstergenin zaman ekseni üzerindeki kaydırma değeri (çubuk bazında).
Method	Ortalama yöntemi.
Applied	Göstergenin hesaplanmasında kullanılan fiyat serisi.

Intraday Time Filter Sinyalleri

Bu modül, piyasa modellerinin etkinliğinin zamanla değiştiği varsayımına dayanır. Bu modülü kullanarak, diğer modüllerden elde edilen sinyalleri saatler ve günlere göre filtreleyebilirsiniz. Arzu edilmeyen zaman-dilimlerinin kesilip atılması yöntemiyle oluşturulan sinyallerin kalitesinin artırılması sağlanır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	Sinyal yok.
Satış için	Sinyal yok.
Alışa itiraz yok	Mevcut tarih ve zaman değeri belirtilen parametrelerle uyuşmakta.
Satışa itiraz yok	Mevcut tarih ve zaman değeri belirtilen parametrelerle uyuşmakta.

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.
GoodHourOfDay	Sinyallerin uygulanacağı saat (0 - 23 arası bir değer). Bu değer -1 ise, herhangi bir saat için herhangi bir sinyal kısıtlaması yapılmaz.
BadHoursOfDay	Bit alanı. Bu alandaki her bir bit, gün içindeki bir saatin değerine karşılık gelir (0 bit - 0 saat, ..., 23 bit - 23-üncü saat). Herhangi bir bit değeri 0 ise, karşılık gelen saat süresince sinyal kısıtlaması yapılmaz. Bit değeri 1 ise, karşılık gelen saat süresince sinyaller engellenecektir. Belirtilen sayı ikili bir sayı olarak temsil edilir ve bit-maskesi olarak kullanılır. Devre dışı bırakılan saatler etkinleştirilen saatlerden daha yüksek önceliğe sahiptirler.
GoodDayOfWeek	Sinyallerin uygulanacağı gün (hafta günü olarak 0 - 6 arası bir değer). Bu değer -1 ise, herhangi bir gün için herhangi bir sinyal kısıtlaması yapılmaz.
BadDaysOfWeek	Bit alanı. Bu alandaki her bir bit, hafta içindeki bir günün değerine karşılık gelir (0 bit - Pazar, ..., 6 bit - Cumartesi). Herhangi bir bit değeri 0 ise, karşılık gelen gün süresince sinyal kısıtlaması yapılmaz. Bit değeri 1 ise, karşılık gelen gün süresince sinyaller engellenecektir. Belirtilen sayı ikili bir sayı olarak temsil edilir ve bit-maskesi olarak kullanılır.

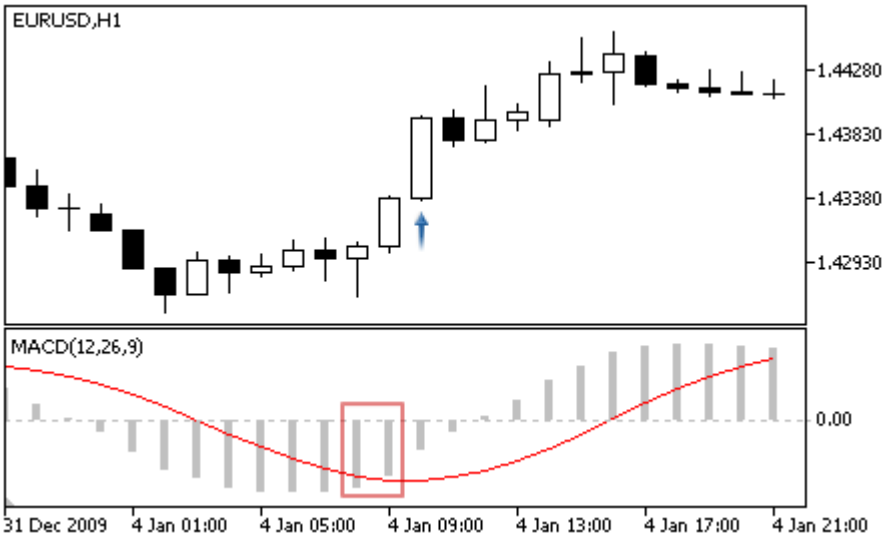
Parameter	Açıklama
	Devre dışı bırakılan günler etkinleştirilen günlerden daha yüksek önceliğe sahiptirler.

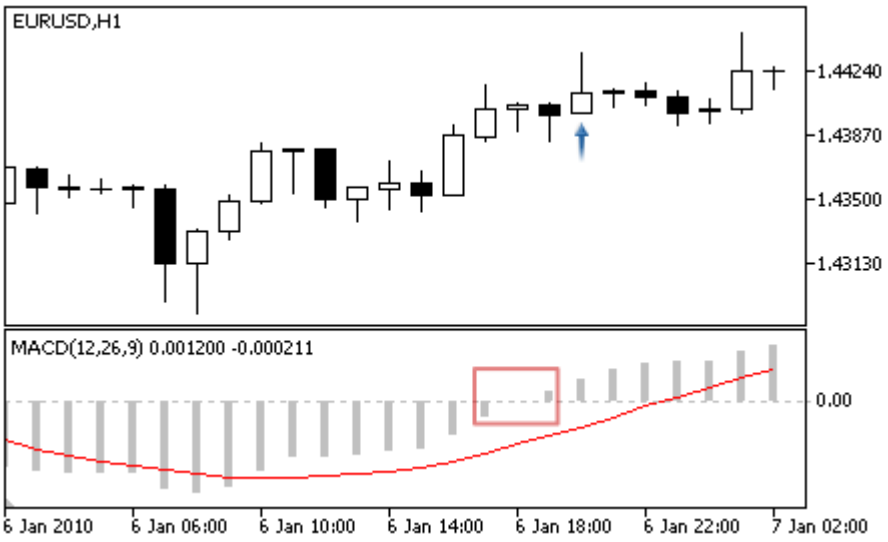
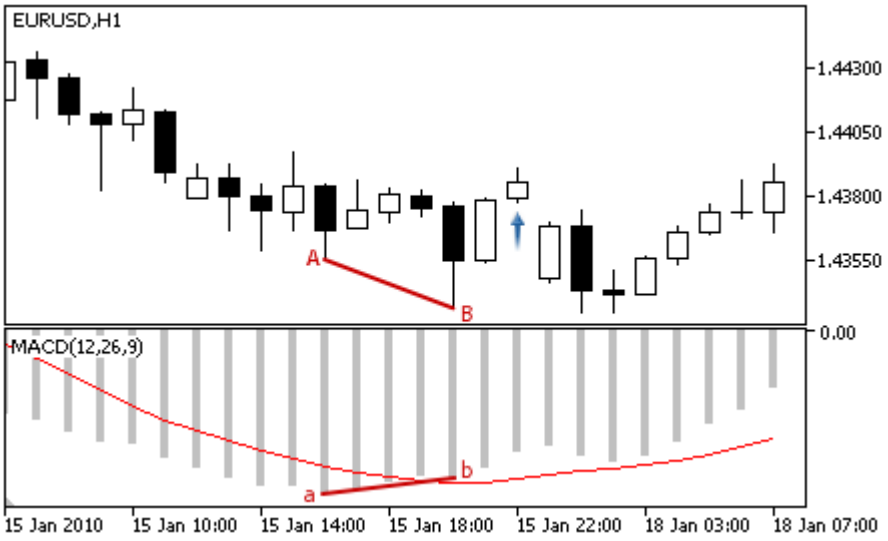
MACD Osilatörünün Sinyalleri

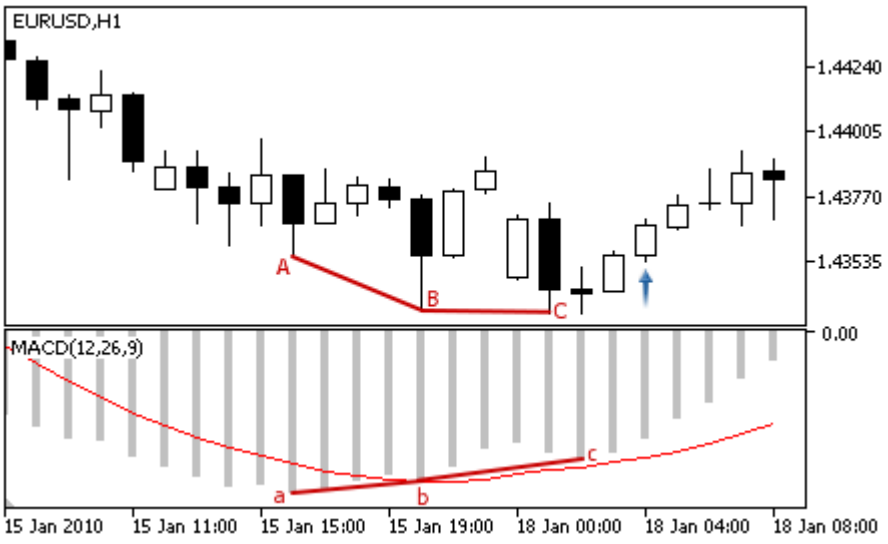
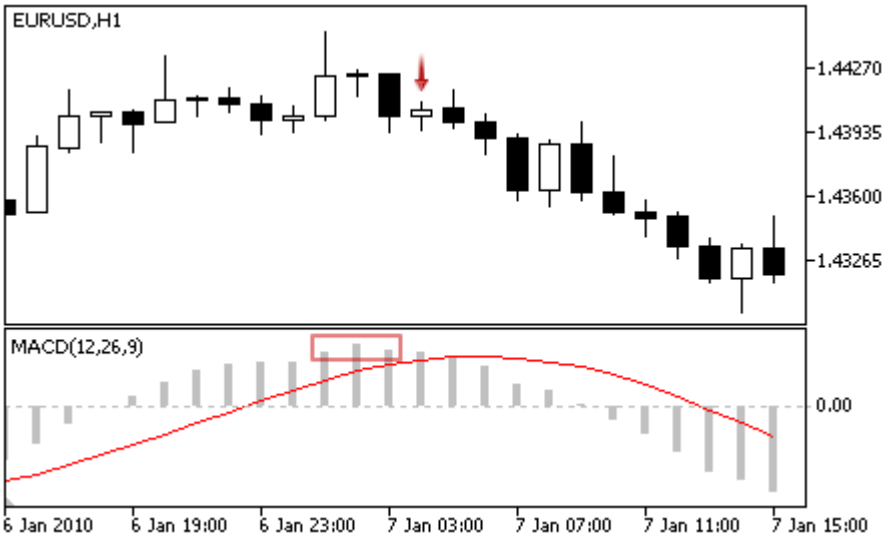
Bu sinyal modülü, MACD osilatörünün piyasa modellerini temel alır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

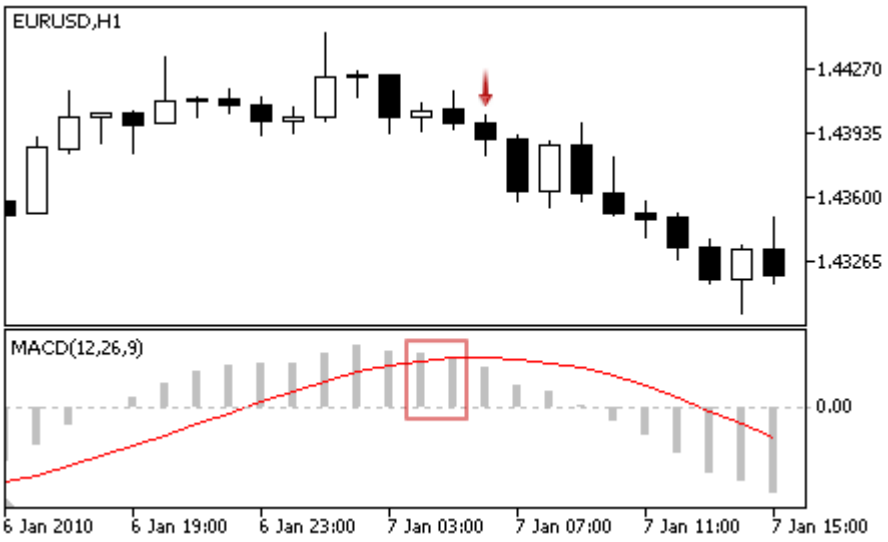
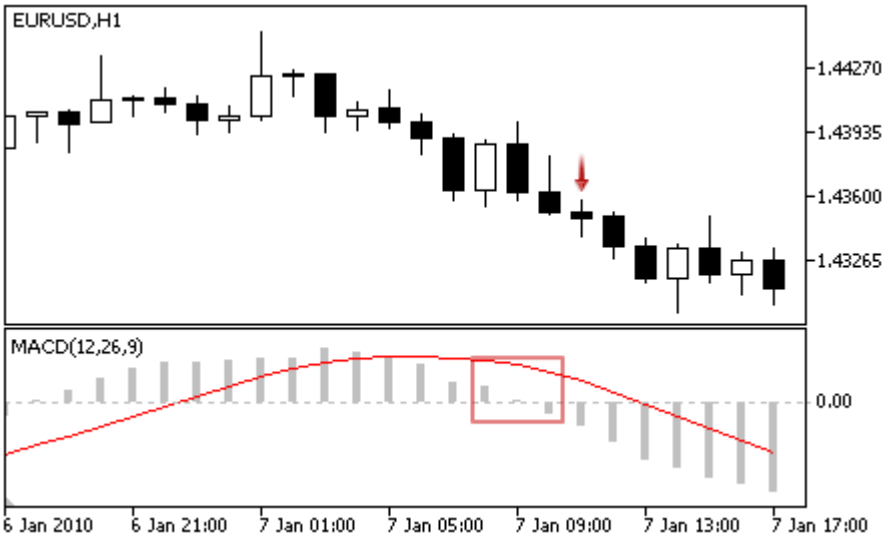
Sinyallerin Oluşma koşulları

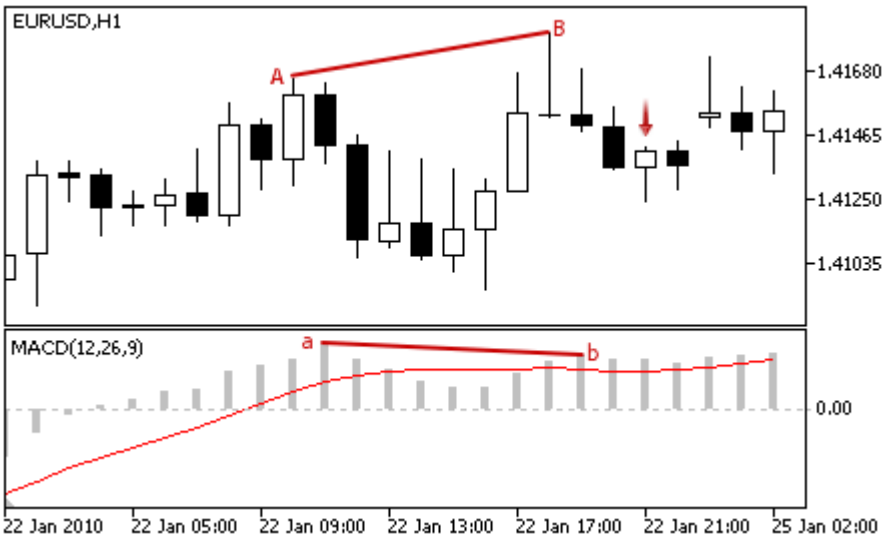
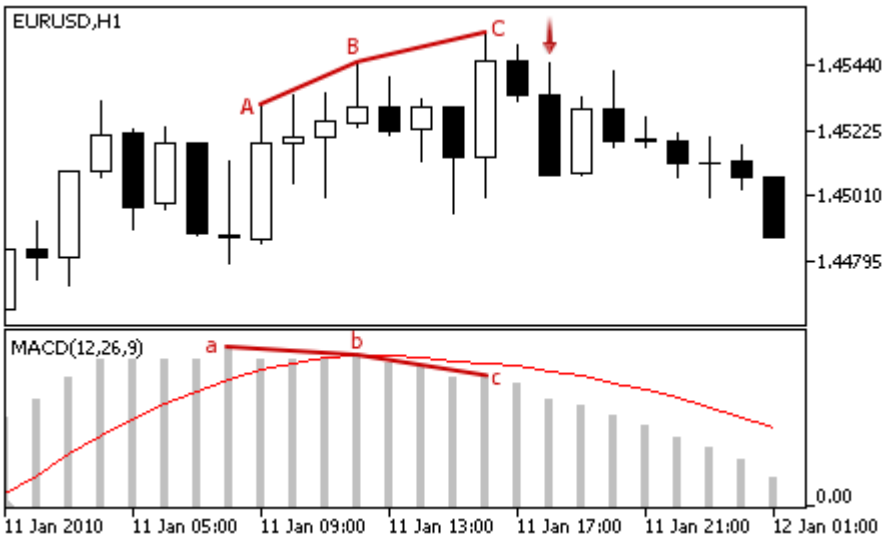
Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	• **Ters** — osilatör yukarı yönelmiştir (osilatör incelenen çubukta yükselirken bir önceki çubukta düşmüştür).  • **Ana-çizgi ile sinyal çizgisinin kesişimi** — ana çizgi incelenen çubukta sinyal çizgisinin üzerindedir ve bir önceki çubukta sinyal çizgisinin altındadır.  • **Sıfır seviyesinin kesilmesi** — ana çizgi incelenen çubukta sıfır çizgisinin üzerindedir ve bir önceki çubukta sıfır çizgisinin altındadır.

Sinyal Tipi	Koşulların Açıklaması
	 • **Ayrışma** — osilatörün analiz edilen ilk tabanı bir öncekinden daha yüksek, ve karşılık gelen fiyat tabanı bir öncekinden daha düşük.  • **Çift ayrışma** — osilatörde üç ardışık taban şekillenmiş, her biri bir öncekinden daha yüksek; aynı şekilde fiyat serisi de üç ardışık tabana sahip ve her biri bir öncekinden daha derin.

Sinyal Tipi	Koşulların Açıklaması
	
Satış için	• **Ters** — osilatör aşağı yönelmiştir (osilatör incelenen çubukta düşerken bir önceki çubukta yükselmiştir).  • **Ana-çizgi ile sinyal çizgisinin kesişimi** — ana çizgi incelenen çubukta sinyal çizgisinin altındadır ve bir önceki çubukta sinyal çizgisinin üzerindedir.

Sinyal Tipi	Koşulların Açıklaması
	 • **Sıfır seviyesinin kesilmesi** — ana çizgi incelenen çubukta sıfır çizgisinin altındadır ve bir önceki çubukta sıfır çizgisinin üzerindedir.  • **Ayrışma** — osilatörün analiz edilen ilk zirvesi bir öncekinden daha düşük, ve karşılık gelen fiyat zirvesi bir öncekinden daha yüksek.

Sinyal Tipi	Koşulların Açıklaması
	 • **Çift ayrışma** — osilatörde üç ardışık zirve şekillenmiş, her biri bir öncekinden daha düşük; aynı şekilde fiyat serisi de üç ardışık zirveye sahip ve her biri bir öncekinden daha yüksek. 
Alışa itiraz yok	Osilatör değeri incelenen çubuk üzerinde artıyor.
Satışa itiraz yok	Osilatör değeri incelenen çubuk üzerinde düşüyor.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

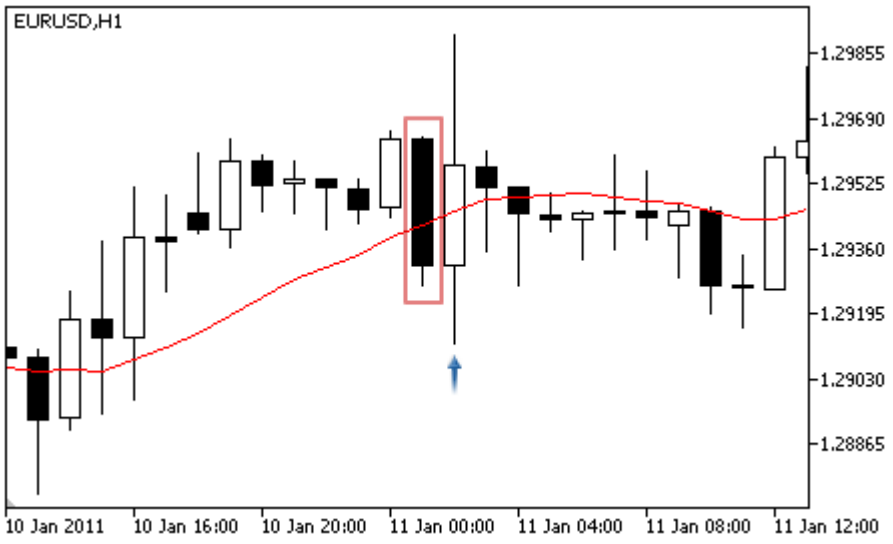
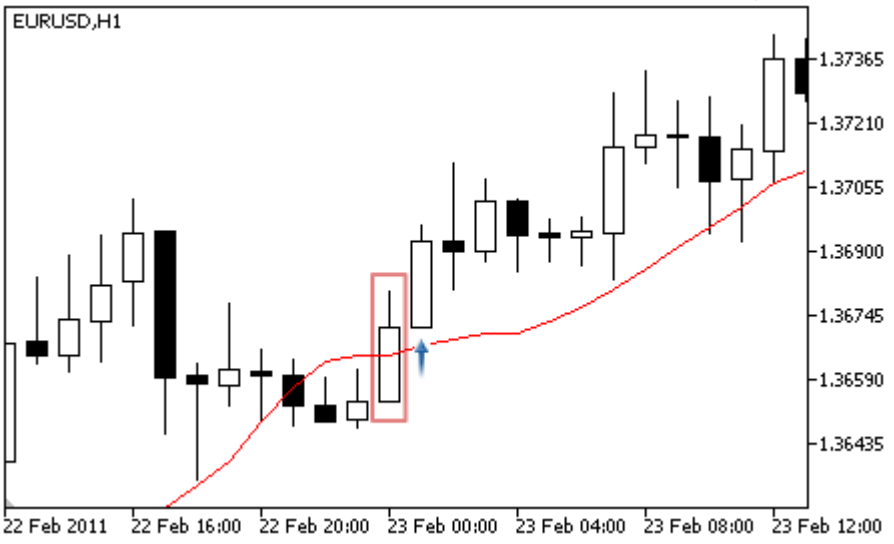
Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.
PeriodFast	Hızlı EMA için hesaplama periyodu.
PeriodSlow	Yavaş EMA için hesaplama periyodu.
PeriodSignal	Düzleştirme periyodu.
Applied	Osilatörün hesaplanmasında kullanılan fiyat serileri.

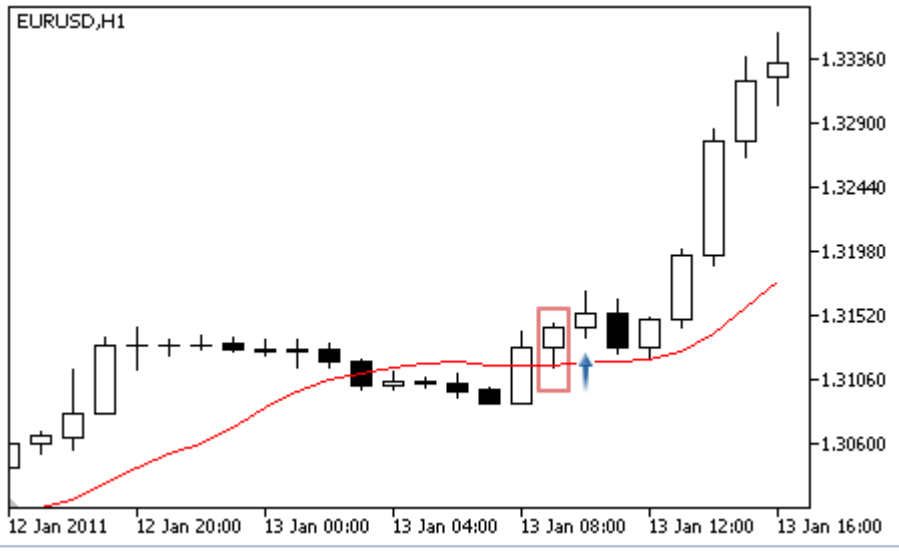
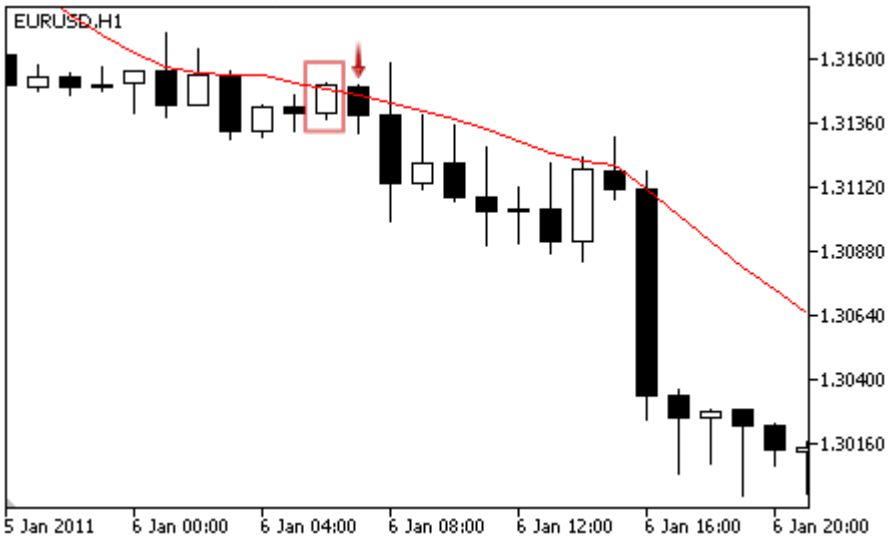
Moving Average Göstergesinin Sinyalleri

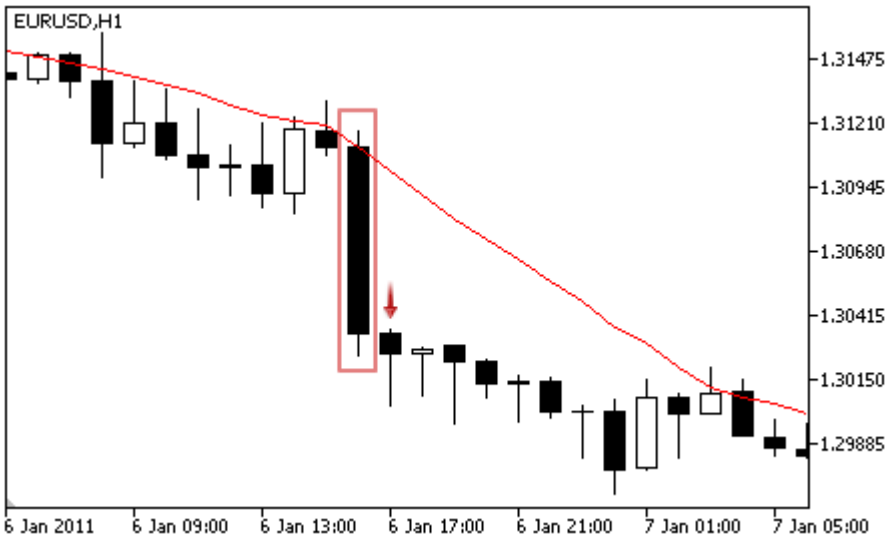
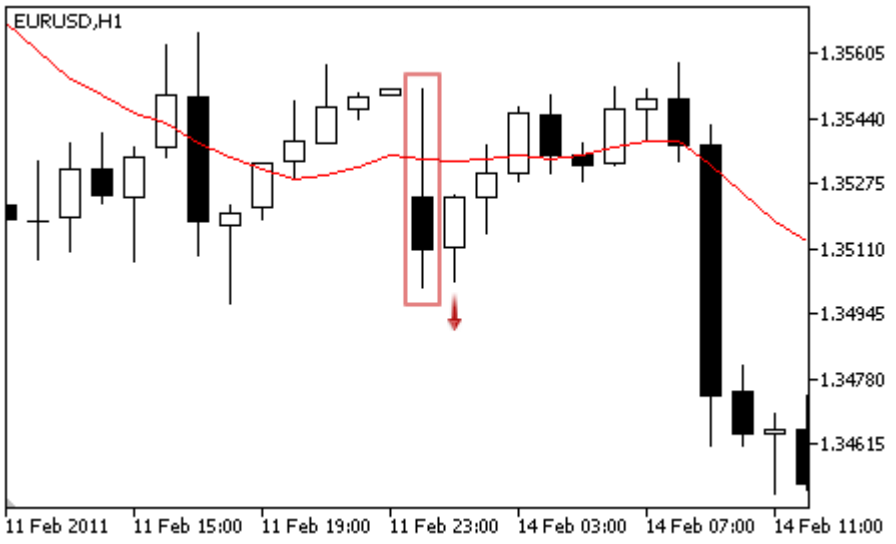
Bu sinyal modülü Moving Average göstergesinin piyasa modellerini temel alır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	• Fiyat serisi göstergeyi aşağı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin üzerinde ve kapanış fiyatı (Close) göstergenin altında) ve gösterge yükseliyor (zayıf sinyal).  • **Hareketli Ortalama kesişimi.** Fiyat serisi göstergeyi yukarı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin altında ve kapanış fiyatı (Close) göstergenin üzerinde) ve gösterge yükseliyor (güçlü sinyal). 

Sinyal Tipi	Koşulların Açıklaması
	• Çubuğun alt gölgesi göstergeyi kesmiş (mevcut çubuğun açılış (Open) ve kapanış (Close) fiyatları göstergenin üzerinde, ve düşük (Low) fiyat göstergenin altında) ve gösterge yükseliyor (zayıf sinyal). 
Satış için	• Fiyat serisi göstergeyi yukarı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin altında ve kapanış fiyatı (Close) göstergenin üzerinde) ve gösterge değeri düşüyor (zayıf sinyal).  • **Hareketli Ortalama kesişimi.** Fiyat serisi göstergeyi aşağı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin üstünde ve kapanış fiyatı (Close) göstergenin altında) ve gösterge değeri düşüyor (güçlü sinyal).

Sinyal Tipi	Koşulların Açıklaması
	 • Çubuğun üst gölgesi göstergeyi kesmiş (mevcut çubuğun açılış (Open) ve kapanış (Close) fiyatları göstergenin altında, ve yüksek (High) fiyat göstergenin üzerinde) ve gösterge düşüyor (zayıf sinyal). 
Alışa itiraz yok	Fiyat göstergenin üzerinde.
Satışa itiraz yok	Fiyat göstergenin altında.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

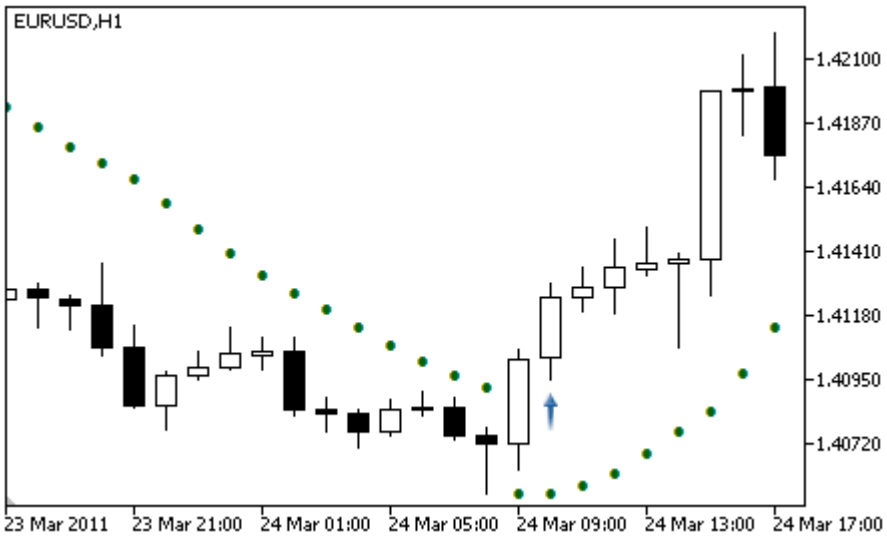
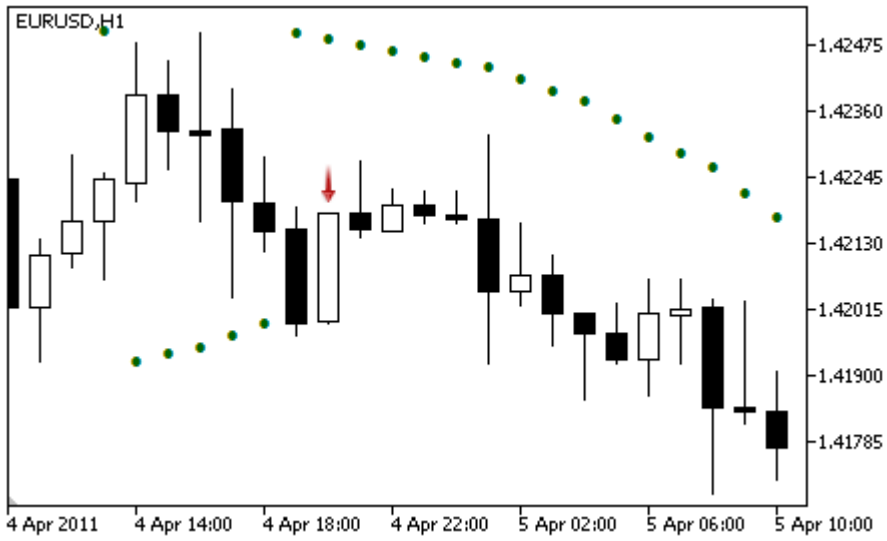
Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.
PeriodMA	Göstergenin hareketli ortalama periyodu.
Shift	Göstergenin zaman ekseni üzerindeki kaydırma değeri (çubuk bazında).
Method	Ortalama yöntemi.
Applied	Göstergenin hesaplanmasında kullanılan fiyat serisi.

Parabolic SAR Göstergesinin Sinyalleri

Bu sinyal modülü Parabolic SAR göstergesinin piyasa modellerini temel alır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	**Ters** — gösterge incelenen çubukta fiyatın altında ve önceki çubukta fiyatın üstündedir. 
Satış için	**Ters** — gösterge incelenen çubukta fiyatın üstünde ve önceki çubukta fiyatın altındadır. 
Alışa itiraz yok	Fiyat göstergenin üzerinde.
Satışa itiraz yok	Fiyat göstergenin altında.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

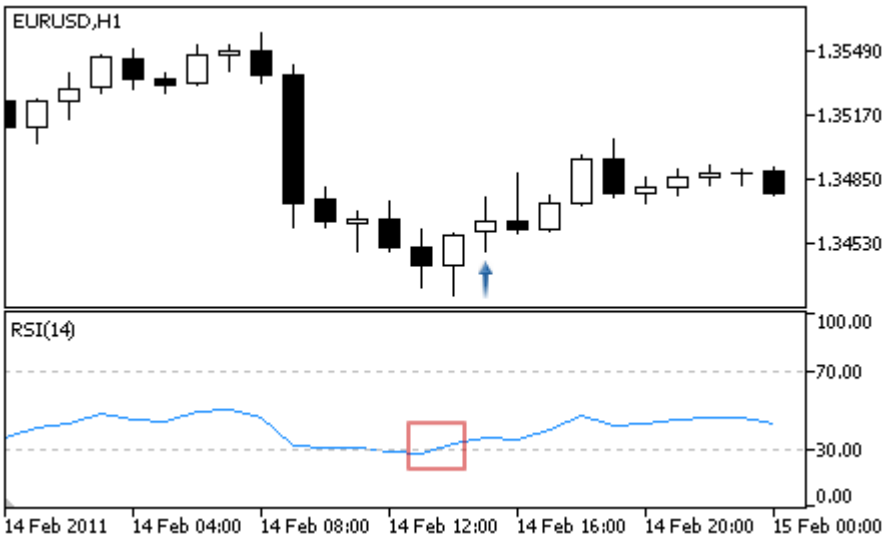
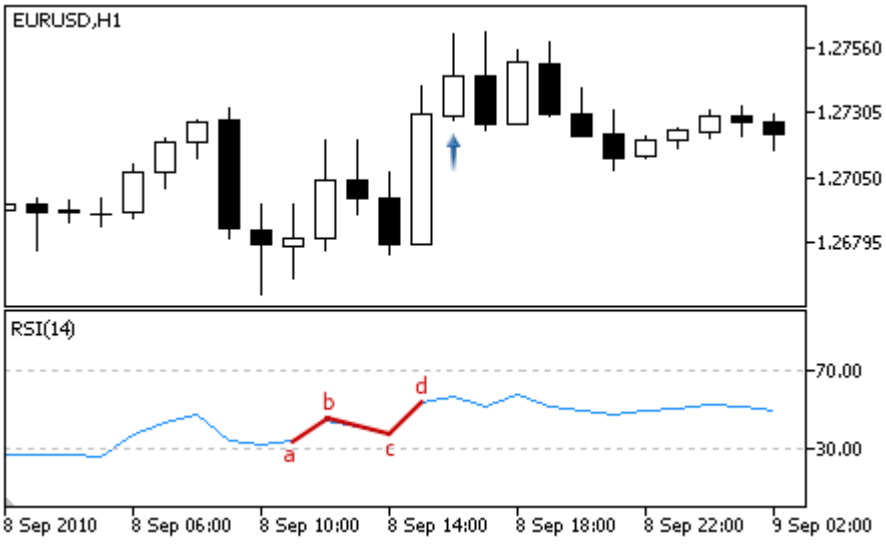
Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.
Step	Göstergenin hız artışı.
Maximum	Gösterge ve fiyat serisi arasındaki maksimum yakınsama hızı.

Relative Strength Index Osilatörünün Sinyalleri

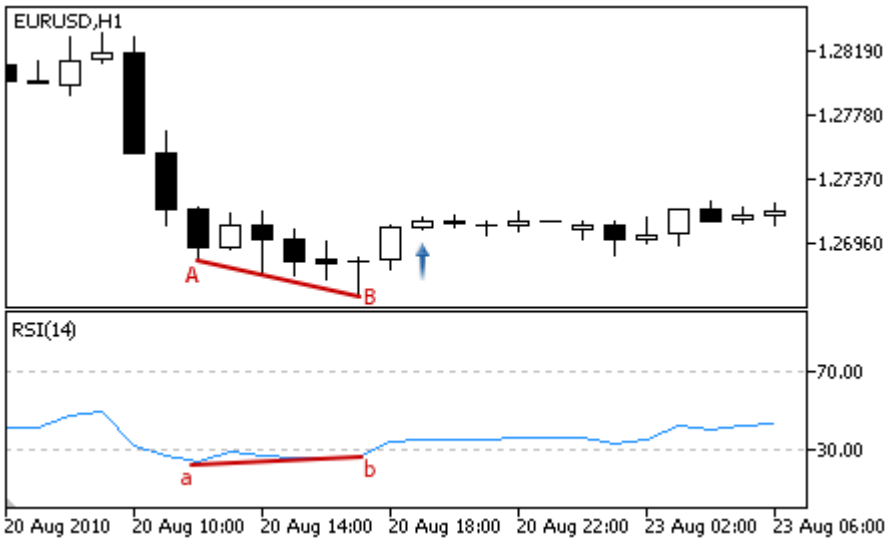
Bu sinyal modülü, Relative Strength Index osilatörünün piyasa modellerini temel alır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

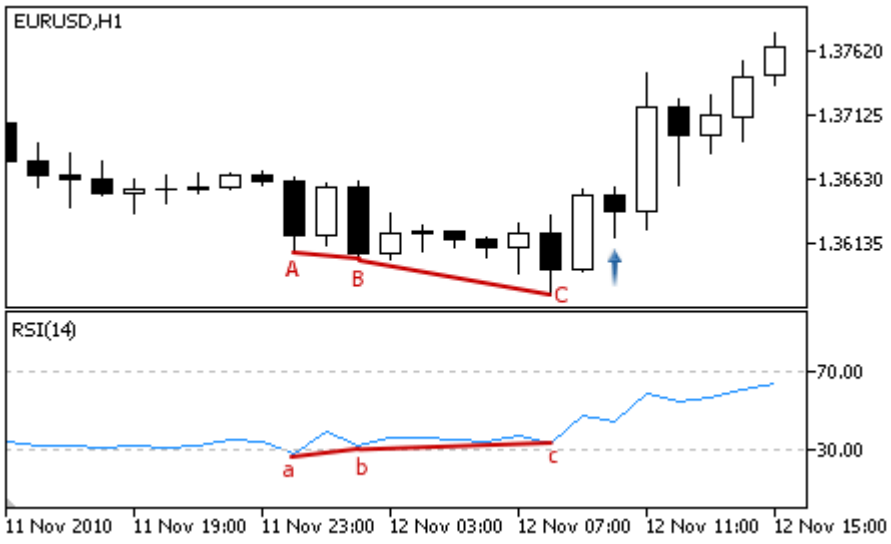
Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	• **Aşırı-satış seviyesinden ters dönüş** — osilatör yukarı yönelmiş ve incelenen mevcut çubuk üzerindeki değeri aşırı-satış seviyesini geçmiş (varsayılan değer : 30).  • **Hatalı dönüş** — incelenen çubuk üzerinde osilatör değeri bir önceki zirveden daha yukarı çıkmıştır.  • **Ayrışma** — osilatörün analiz edilen ilk tabanı bir öncekinden daha sığ, ve karşılık gelen fiyat tabanı bir öncekinden daha derin.

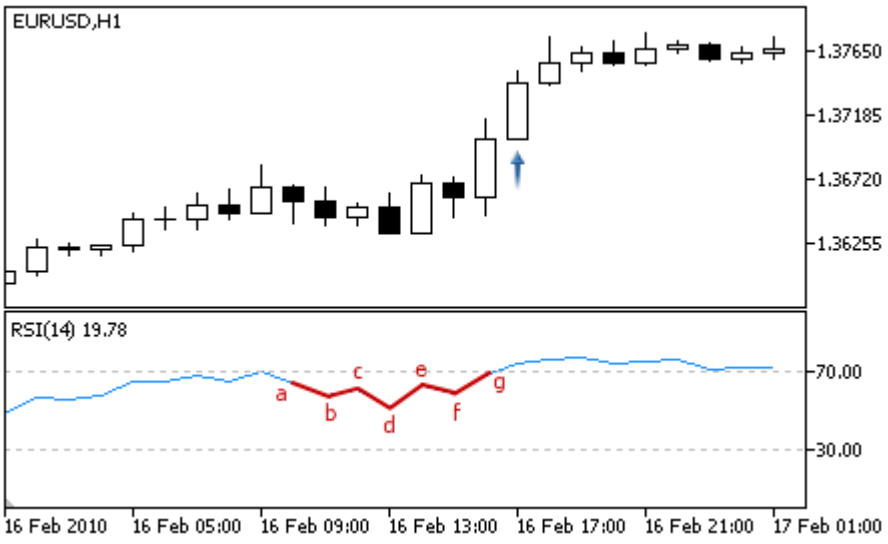
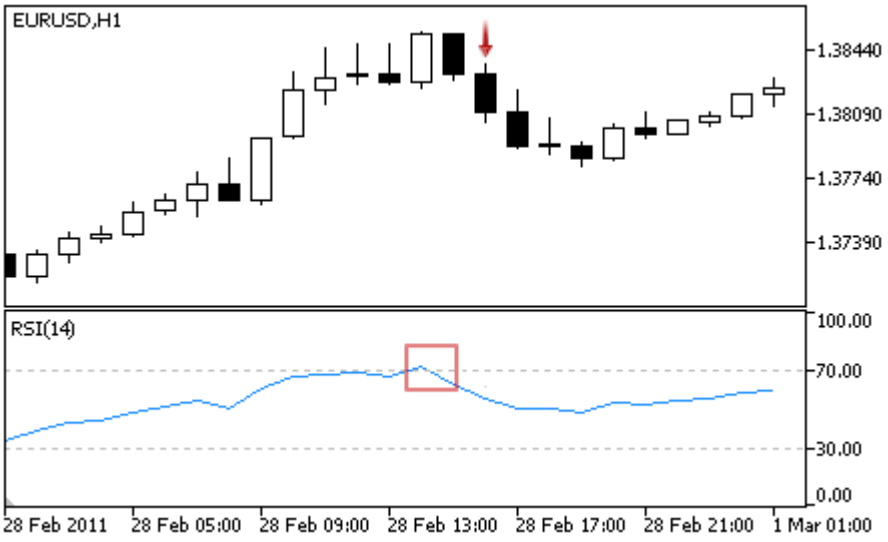
Sinyal Tipi	Koşulların Açıklaması

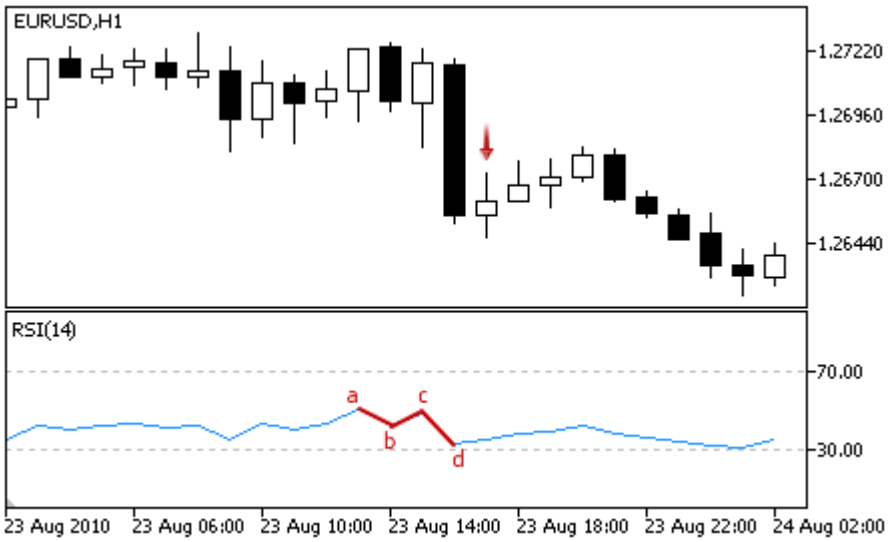
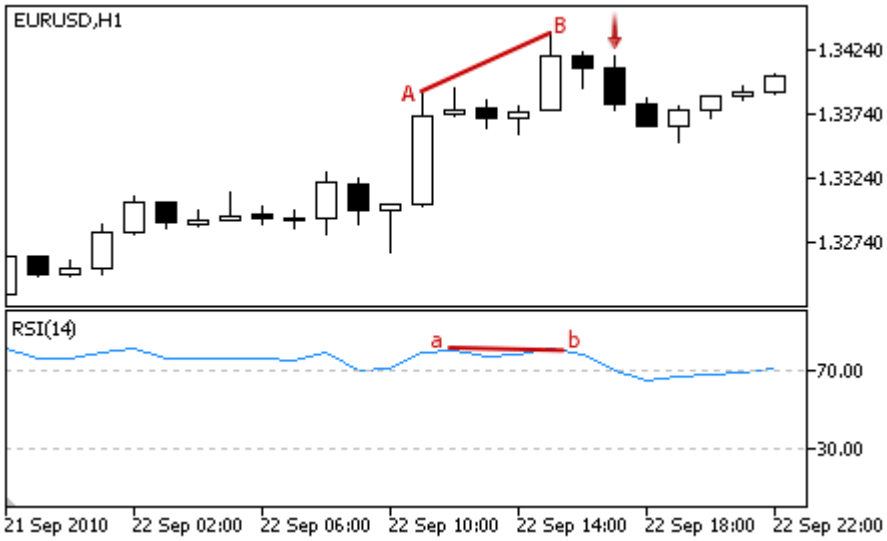


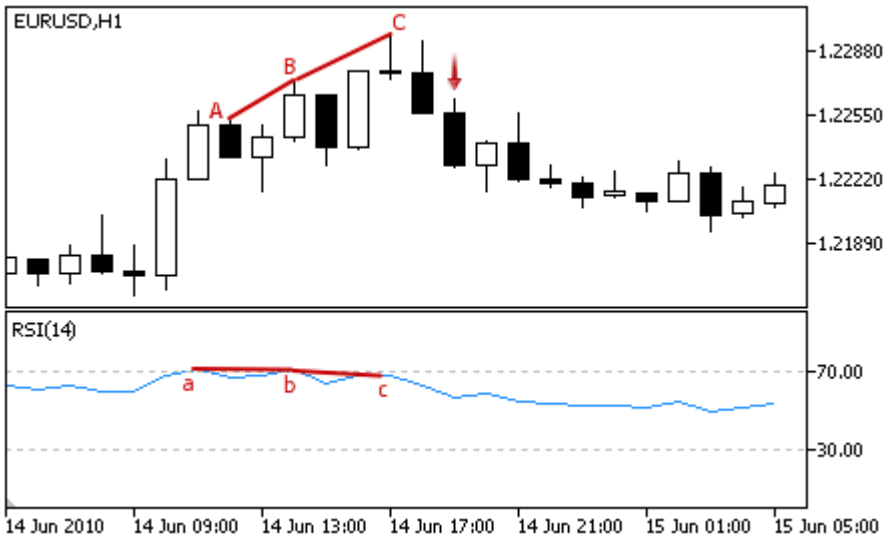
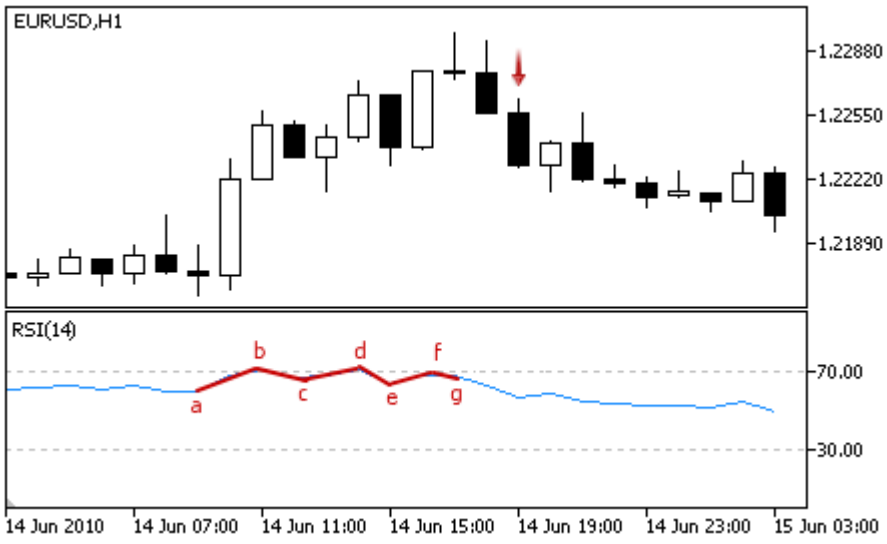
- **Çift ayrışma** — osilatörde üç ardışık taban şekillenmiş, her biri bir öncekinden daha yüksek; aynı şekilde fiyat serisi de üç ardışık tabana sahip ve her biri bir öncekinden daha derin.



- **Baş/Omuz** — osilatörde üç ardışık taban şekillenmiştir ve ortadaki diğerlerinden daha alçaktır.

Sinyal Tipi	Koşulların Açıklaması
	
Satış için	• **Aşırı-alış seviyesinden ters dönüş** — osilatör aşağı yönelmiş ve incelenen mevcut çubuk üzerindeki değeri aşırı-alış seviyesini geçmiş (varsayılan değer : 70).  • **Hatalı dönüş** — incelenen çubuk üzerinde osilatör değeri bir önceki tabandan daha aşağı düşmüştür.

Sinyal Tipi	Koşulların Açıklaması
	 • **Ayrışma** — osilatörün analiz edilen ilk zirvesi bir öncekinden daha düşük, ve karşılık gelen fiyat zirvesi bir öncekinden daha yüksek.  • **Çift ayrışma** — osilatörde üç ardışık zirve şekillenmiş, her biri bir öncekinden daha düşük; aynı şekilde fiyat serisi de üç ardışık zirveye sahip ve her biri bir öncekinden daha yüksek.

Sinyal Tipi	Koşulların Açıklaması
	 • **Baş/Omuz** — osilatörde üç ardışık zirve şekillenmiştir ve ortadaki diğerlerinden daha yüksektir. 
Alışa itiraz yok	Osilatör değeri incelenen çubuk üzerinde artıyor.
Satışa itiraz yok	Osilatör değeri incelenen çubuk üzerinde düşüyor.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

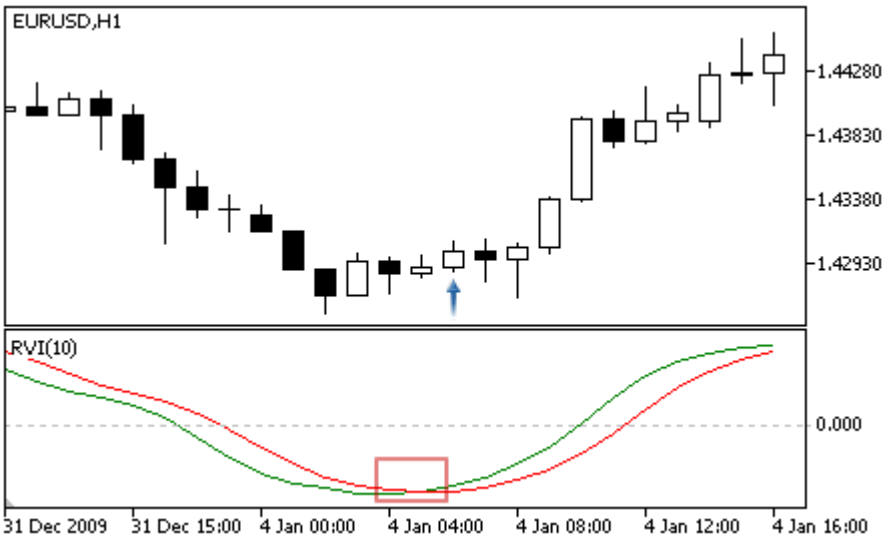
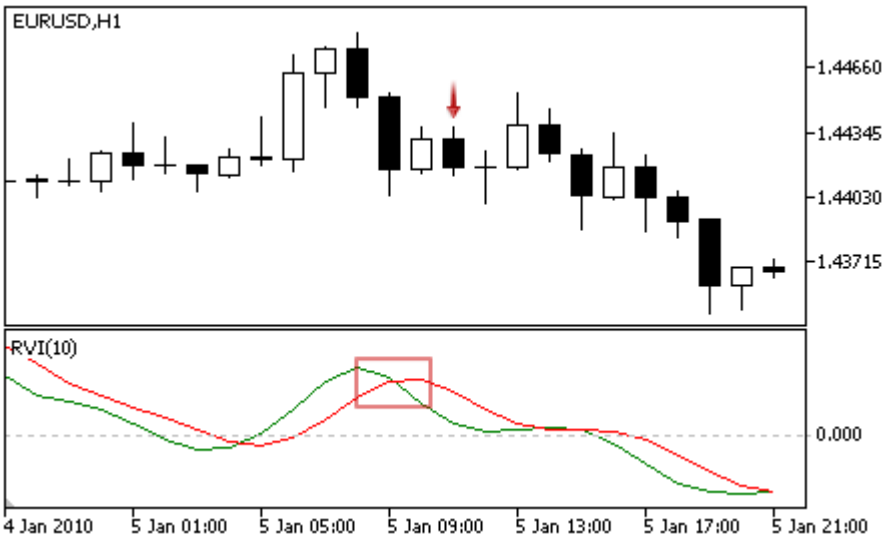
Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.
PeriodRSI	Osilatörün hesaplanmasında kullanılan periyot değeri.
Applied	Osilatörün hesaplanmasında kullanılan fiyat serileri.

Relative Vigor Index Osilatörünün Sinyalleri

Bu sinyal modülü Relative Vigor Index osilatörünün piyasa modellerini temel alır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	**Ana-çizgi ile sinyal çizgisinin kesişimi** — ana çizgi incelenen çubukta sinyal çizgisinin üzerindedir ve bir önceki çubukta sinyal çizgisinin altındadır. 
Satış için	**Ana-çizgi ile sinyal çizgisinin kesişimi** — ana çizgi incelenen çubukta sinyal çizgisinin altındadır ve bir önceki çubukta sinyal çizgisinin üzerindedir. 
Alışa itiraz yok	Osilatör değeri incelenen çubuk üzerinde artıyor.

Sinyal Tipi	Koşulların Açıklaması
Satışa itiraz yok	Osilatör değeri incelenen çubuk üzerinde düşüyor.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

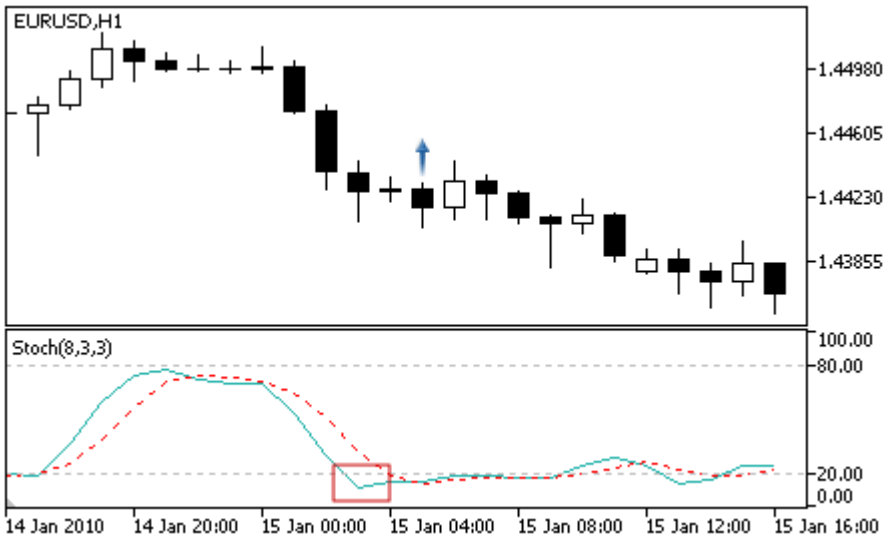
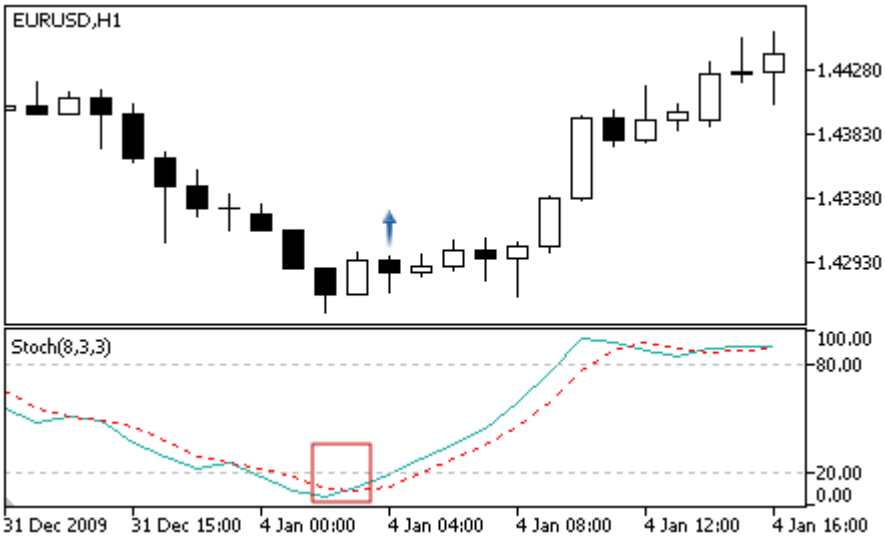
Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.
PeriodRVI	Osilatörün hesaplanmasında kullanılan periyot değeri.

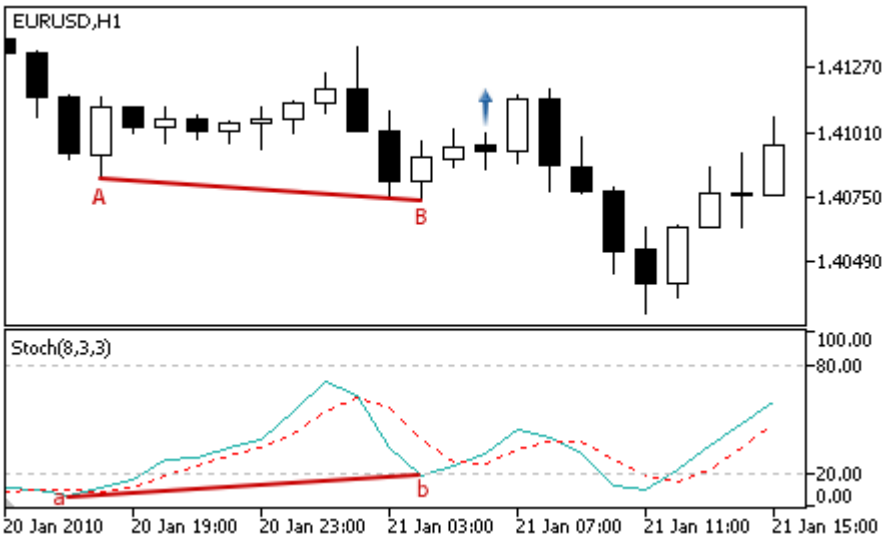
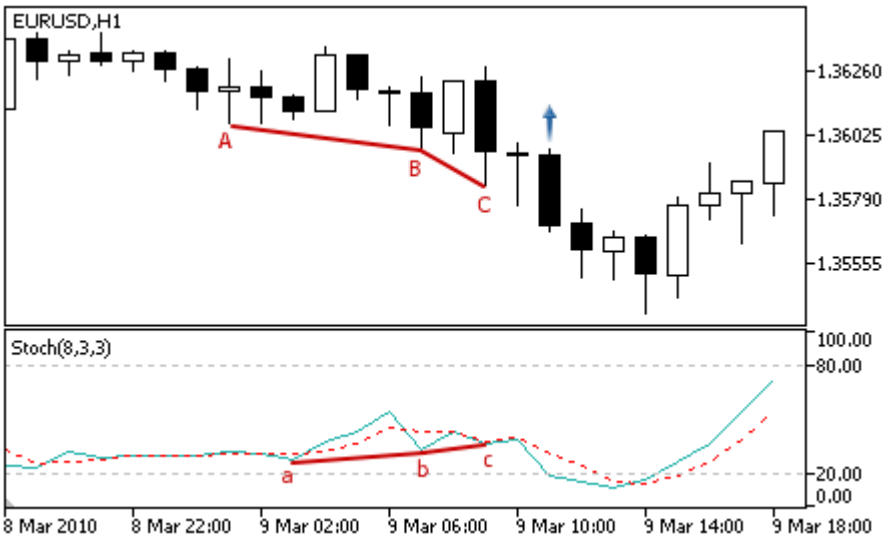
Stochastic Osilatörünün Sinyalleri

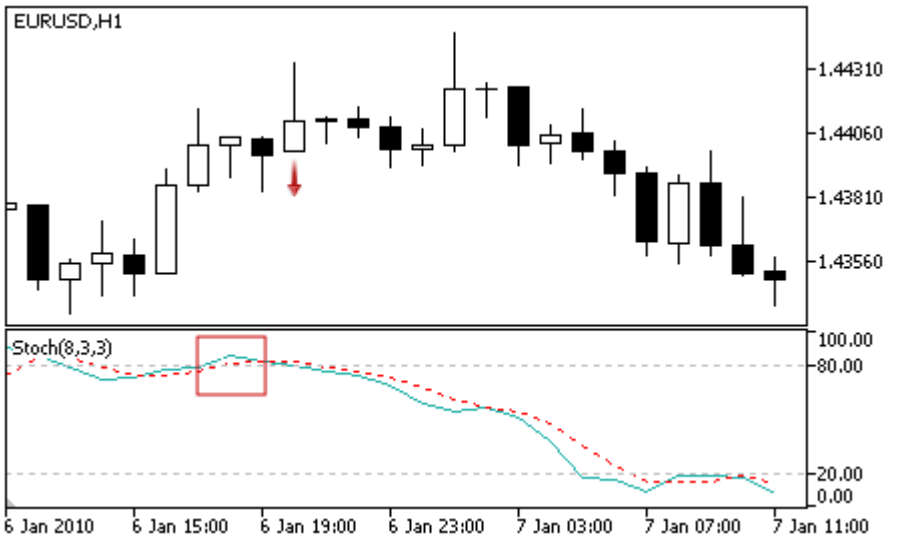
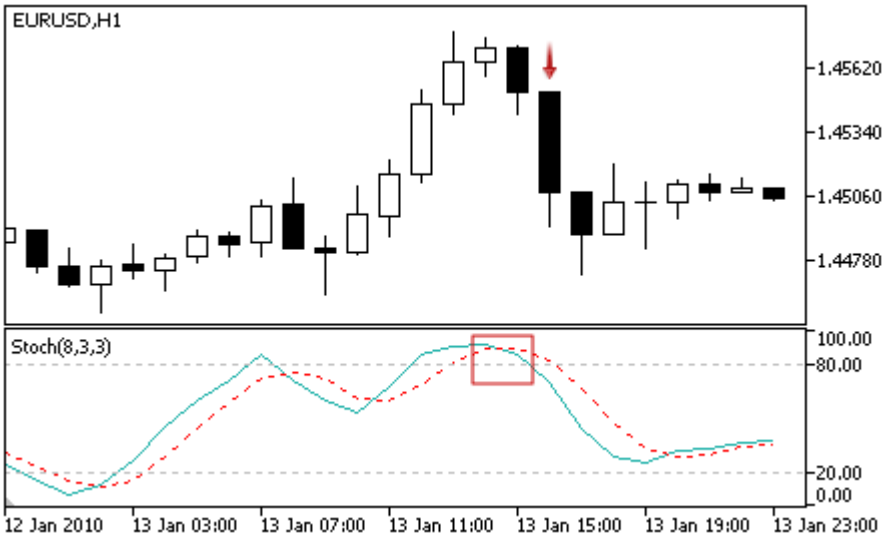
Bu sinyal modülü Stochastic göstergesinin piyasa modellerine dayanır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

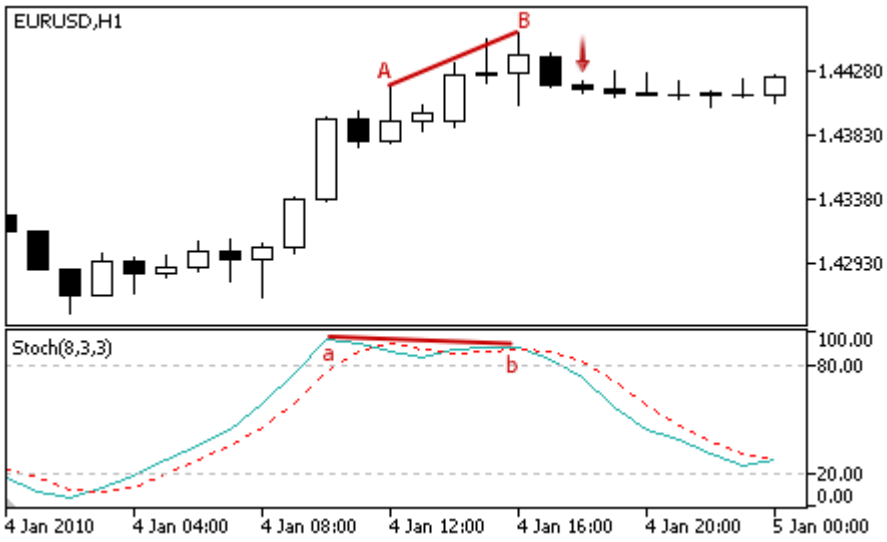
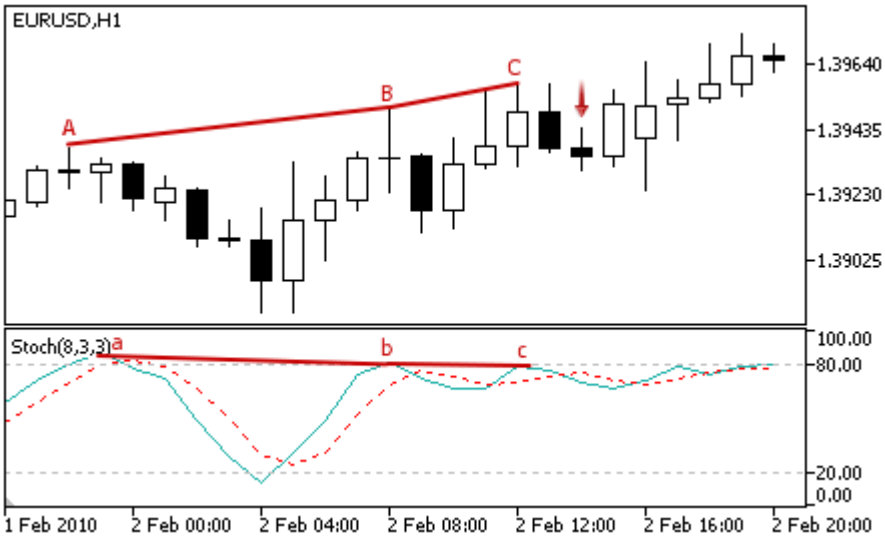
Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	• **Ters** — osilatör yukarı yönelmiştir (osilatör incelenen çubukta yükselirken bir önceki çubukta düşmüştür).  • **Ana-çizgi ile sinyal çizgisinin kesişimi** — ana çizgi incelenen çubukta sinyal çizgisinin üzerindedir ve bir önceki çubukta sinyal çizgisinin altındadır.  • **Ayrışma** — osilatörün analiz edilen ilk tabanı bir öncekinden daha yüksek, ve karşılık gelen fiyat tabanı bir öncekinden daha düşük.

Sinyal Tipi	Koşulların Açıklaması
	 • **Çift ayrışma** — osilatörde üç ardışık taban şekillenmiş, her biri bir öncekinden daha yüksek; aynı şekilde fiyat serisi de üç ardışık tabana sahip ve her biri bir öncekinden daha derin. 
Satış için	• **Ters** — osilatör aşağı yönelmiştir (osilatör incelenen çubukta düşerken bir önceki çubukta yükselmiştir).

Sinyal Tipi	Koşulların Açıklaması
	 • **Ana-çizgi ile sinyal çizgisinin kesişimi** — ana çizgi incelenen çubukta sinyal çizgisinin altındadır ve bir önceki çubukta sinyal çizgisinin üzerindedir.  • **Ayrışma** — osilatörün analiz edilen ilk zirvesi bir öncekinden daha düşük, ve karşılık gelen fiyat zirvesi bir öncekinden daha yüksek.

Sinyal Tipi	Koşulların Açıklaması
	 • **Çift ayrışma** — osilatörde üç ardışık zirve şekillenmiş, her biri bir öncekinden daha düşük; aynı şekilde fiyat serisi de üç ardışık zirveye sahip ve her biri bir öncekinden daha yüksek. 
Alışa itiraz yok	Osilatör değeri incelenen çubuk üzerinde artıyor.
Satışa itiraz yok	Osilatör değeri incelenen çubuk üzerinde düşüyor.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

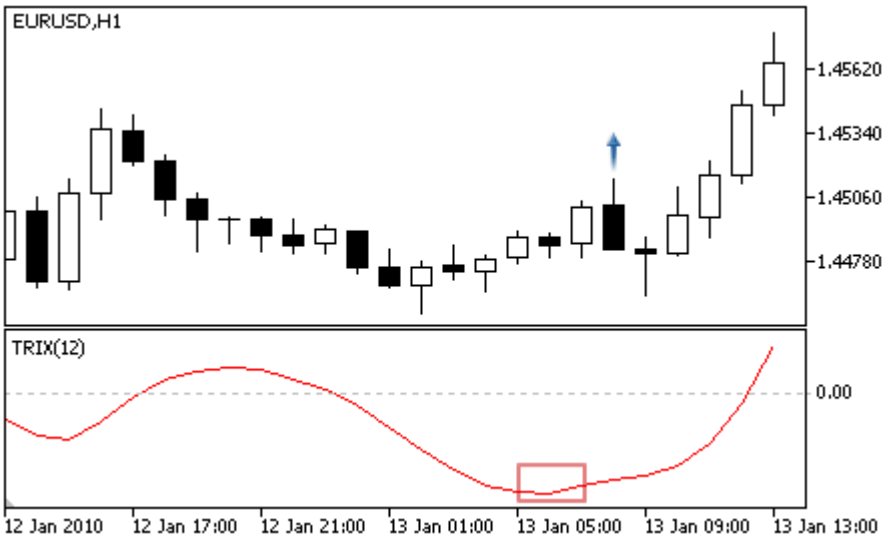
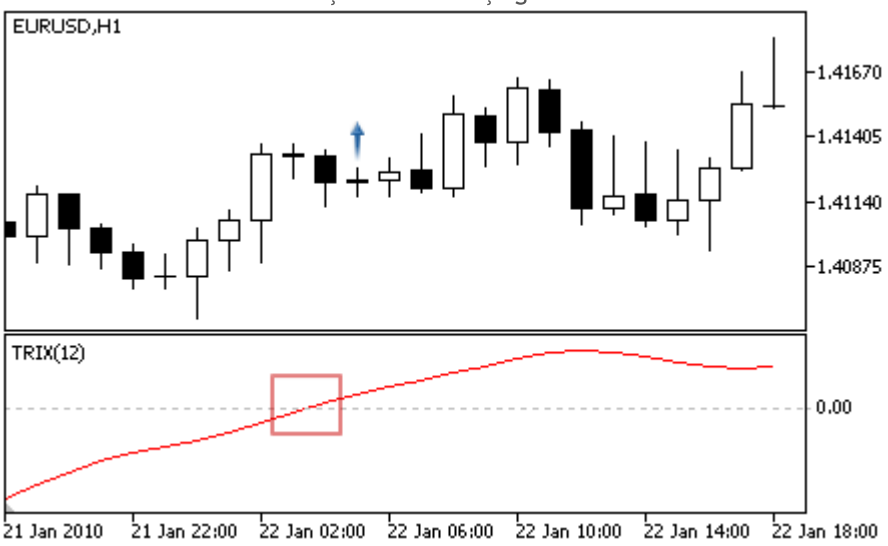
Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.
PeriodK	Osilatörün %K çizgisinin hesaplanmasında kullanılan periyot değeri.
PeriodD	Osilatörün %D çizgisinin hesaplanmasında kullanılan periyot değeri.
PeriodSlow	Yavaşlama periyodu.
Applied	Osilatörün hesaplanmasında kullanılan fiyat serileri.

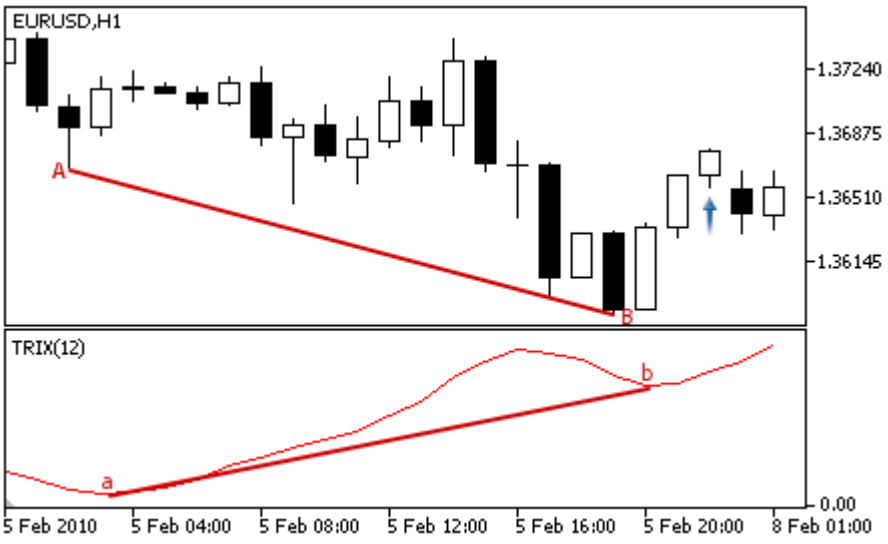
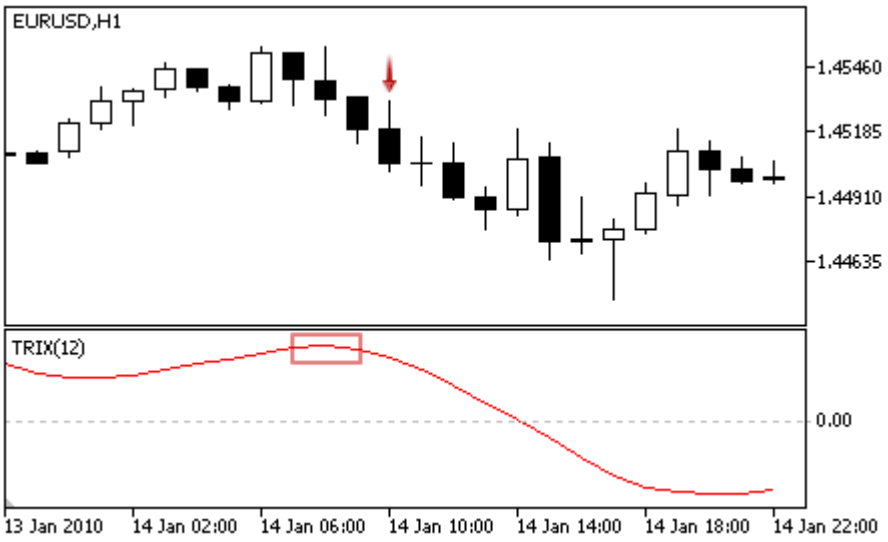
Triple Exponential Average Osilatörünün Sinyalleri

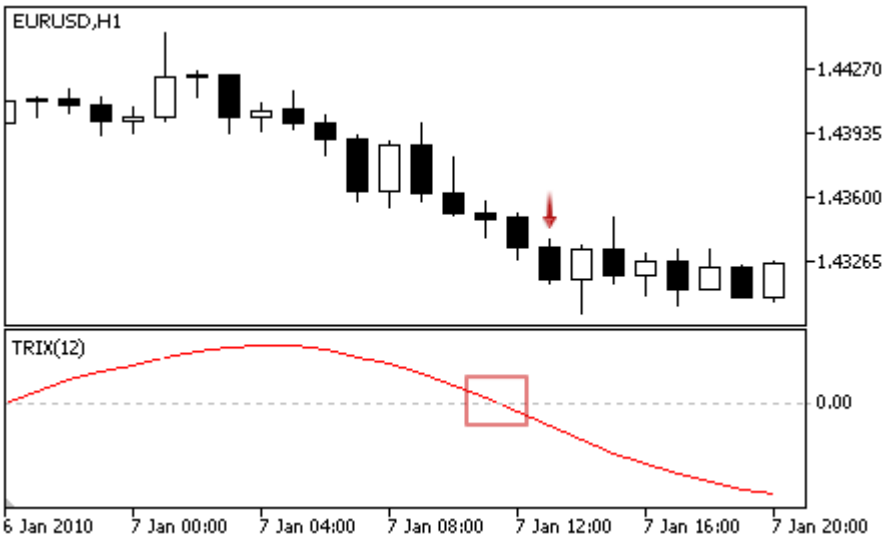
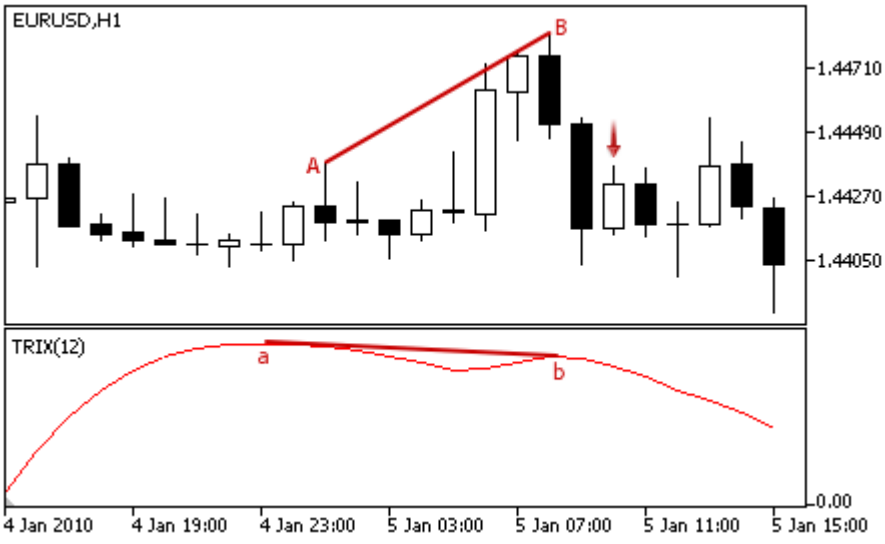
Bu sinyal modülü, Triple Exponential Average göstergesinin piyasa modellerini temel alır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	• **Ters** — osilatör yukarı yönelmiştir (osilatör incelenen çubukta yükselirken bir önceki çubukta düşmüştür).  • **Sıfır seviyesinin kesilmesi** — ana çizgi incelenen çubukta sıfır çizgisinin üzerindedir ve bir önceki çubukta sıfır çizgisinin altındadır.  • **Ayrışma** — osilatörün analiz edilen ilk tabanı bir öncekinden daha yüksek, ve karşılık gelen fiyat tabanı bir öncekinden daha düşük.

Sinyal Tipi	Koşulların Açıklaması
	
Satış için	• **Ters** — osilatör aşağı yönelmiştir (osilatör incelenen çubukta düşerken bir önceki çubukta yükselmiştir).  • **Sıfır seviyesinin kesilmesi** — ana çizgi incelenen çubukta sıfır çizgisinin altındadır ve bir önceki çubukta sıfır çizgisinin üzerindedir.

Sinyal Tipi	Koşulların Açıklaması
	 • **Ayrışma** — osilatörün analiz edilen ilk zirvesi bir öncekinden daha düşük, ve karşılık gelen fiyat zirvesi bir öncekinden daha yüksek. 
Alışa itiraz yok	Osilatör değeri incelenen çubuk üzerinde artıyor.
Satışa itiraz yok	Osilatör değeri incelenen çubuk üzerinde düşüyor.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.

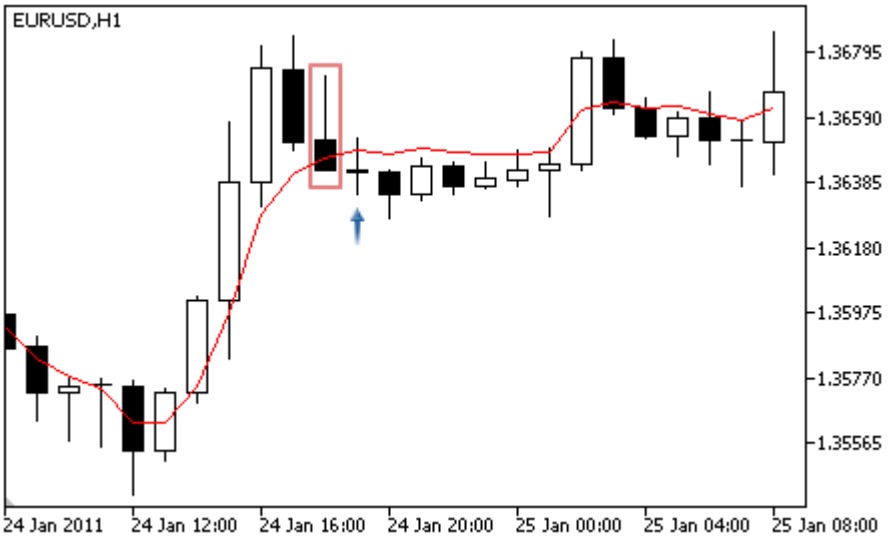
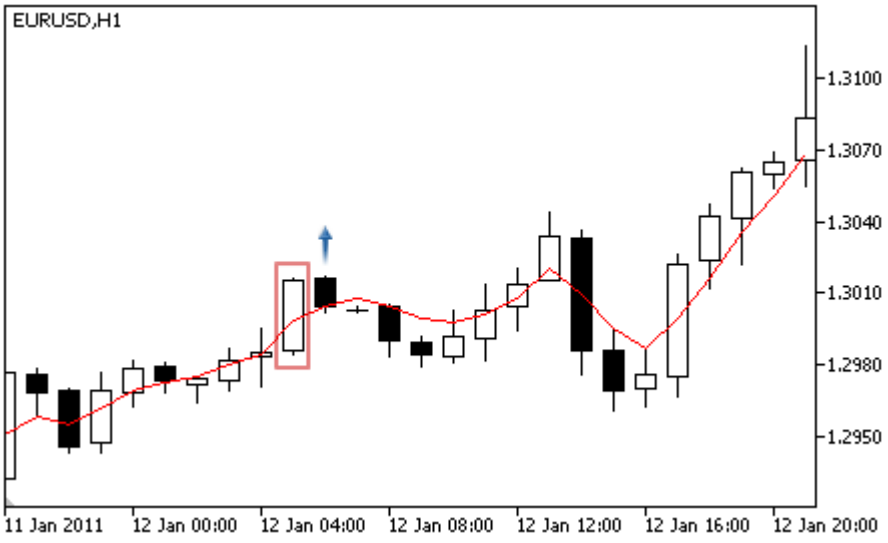
Parameter	Açıklama
PeriodTriX	Osilatörün hesaplanmasında kullanılan periyot değeri.
Applied	Osilatörün hesaplanmasında kullanılan fiyat serileri.

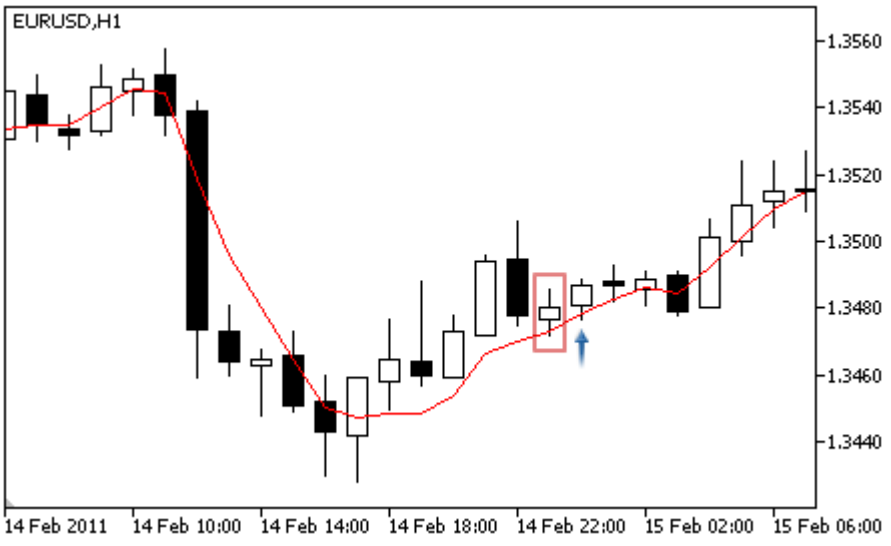
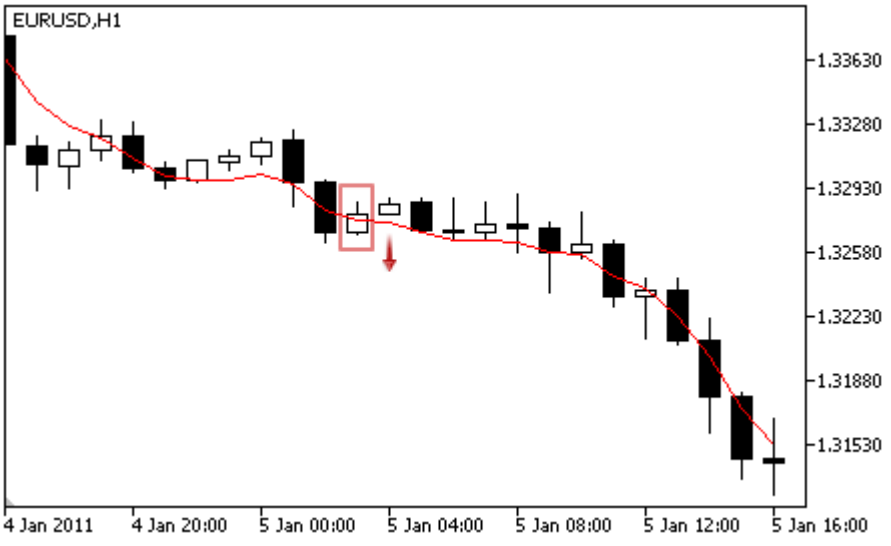
Triple Exponential Moving Average Göstergesinin Sinyalleri

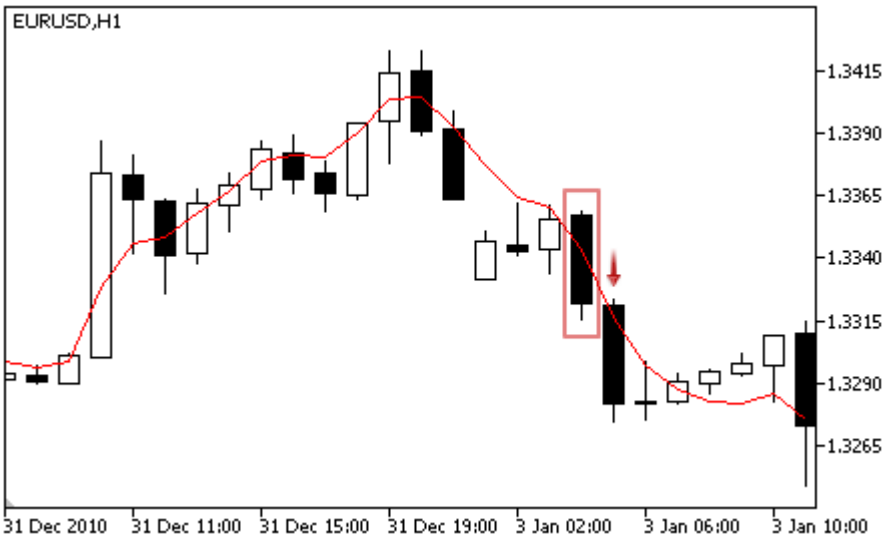
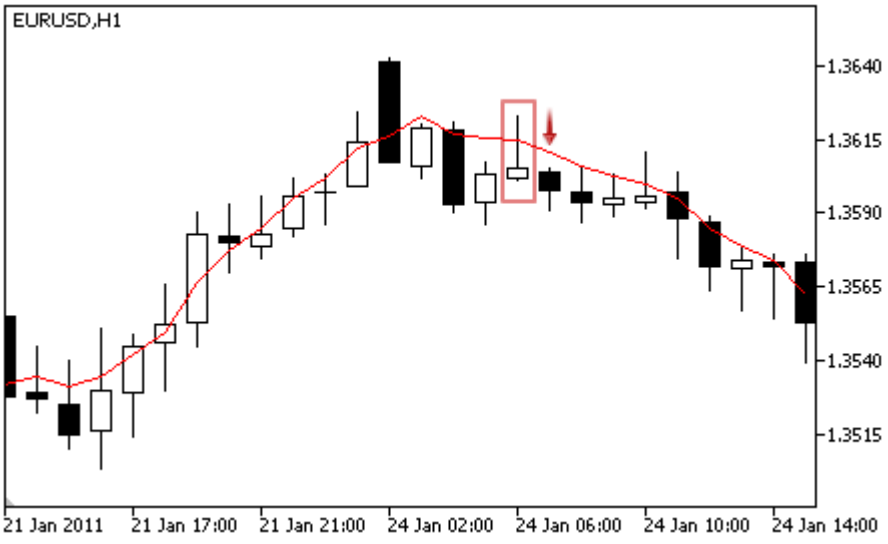
Bu sinyal modülü, Triple Exponential Moving Average göstergesinin piyasa modellerini temel alır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	• Fiyat serisi göstergeyi aşağı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin üzerinde ve kapanış fiyatı (Close) göstergenin altında) ve gösterge yükseliyor (zayıf sinyal).  • **Hareketli Ortalama kesişimi.** Fiyat serisi göstergeyi yukarı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin altında ve kapanış fiyatı (Close) göstergenin üzerinde) ve gösterge yükseliyor (güçlü sinyal). 

Sinyal Tipi	Koşulların Açıklaması
	• Çubuğun alt gölgesi göstergeyi kesmiş (mevcut çubuğun açılış (Open) ve kapanış (Close) fiyatları göstergenin üzerinde, ve düşük (Low) fiyat göstergenin altında) ve gösterge yükseliyor (zayıf sinyal). 
Satış için	• Fiyat serisi göstergeyi yukarı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin altında ve kapanış fiyatı (Close) göstergenin üzerinde) ve gösterge değeri düşüyor (zayıf sinyal).  • **Hareketli Ortalama kesişimi.** Fiyat serisi göstergeyi aşağı doğru kesmiş (incelenen çubuğun açılış fiyatı (Open) göstergenin üstünde ve kapanış fiyatı (Close) göstergenin altında) ve gösterge değeri düşüyor (güçlü sinyal).

Sinyal Tipi	Koşulların Açıklaması
	 • Çubuğun üst gölgesi göstergeyi kesmiş (mevcut çubuğun açılış (Open) ve kapanış (Close) fiyatları göstergenin altında, ve yüksek (High) fiyat göstergenin üzerinde) ve gösterge düşüyor (zayıf sinyal). 
Alışa itiraz yok	Fiyat göstergenin üzerinde.
Satışa itiraz yok	Fiyat göstergenin altında.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

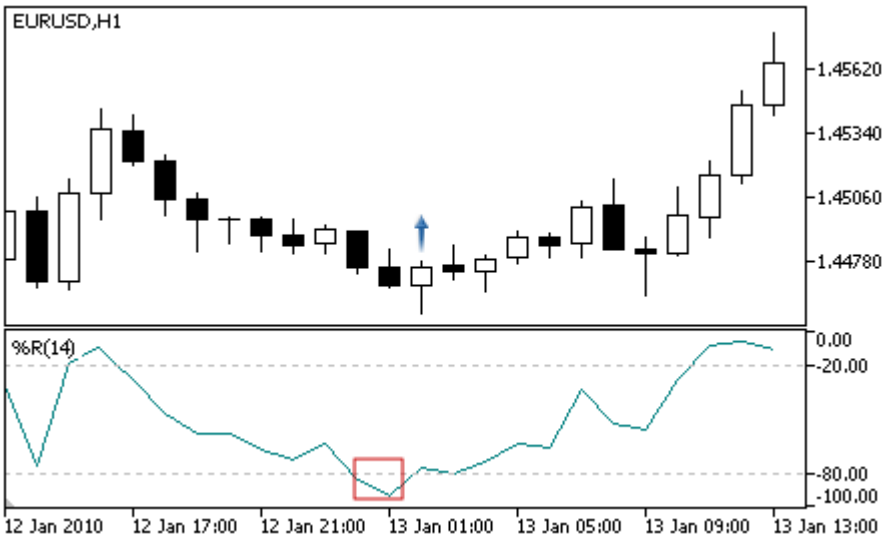
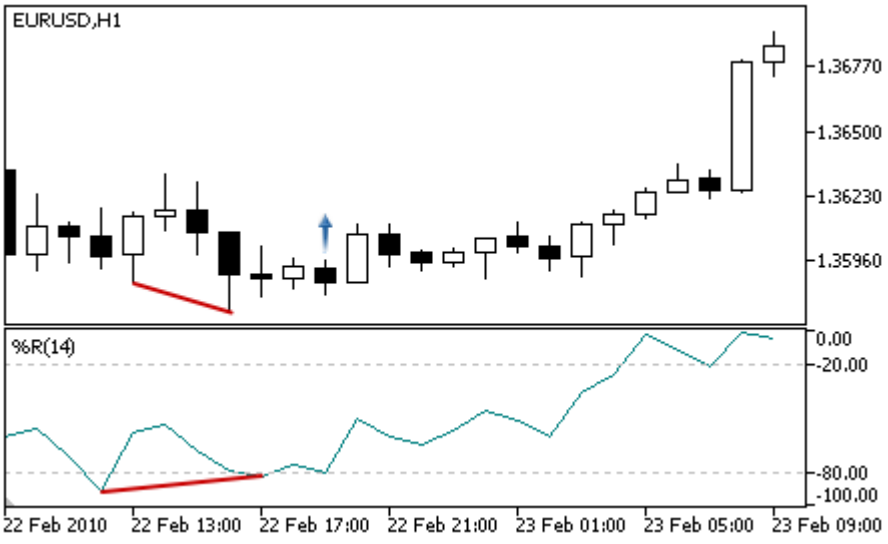
Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.
PeriodMA	Göstergenin hareketli ortalama periyodu.
Shift	Göstergenin zaman ekseni üzerindeki kaydırma değeri (çubuk bazında).
Method	Ortalama yöntemi.
Applied	Göstergenin hesaplanmasında kullanılan fiyat serisi.

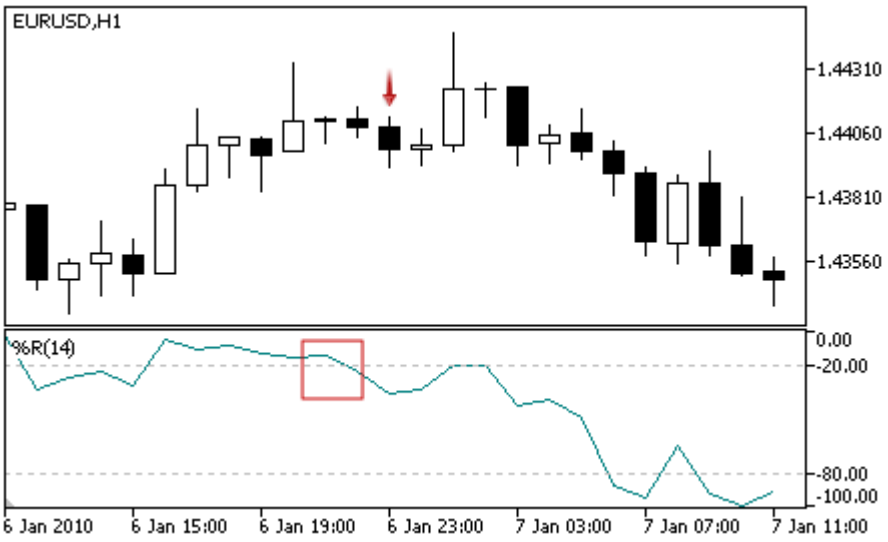
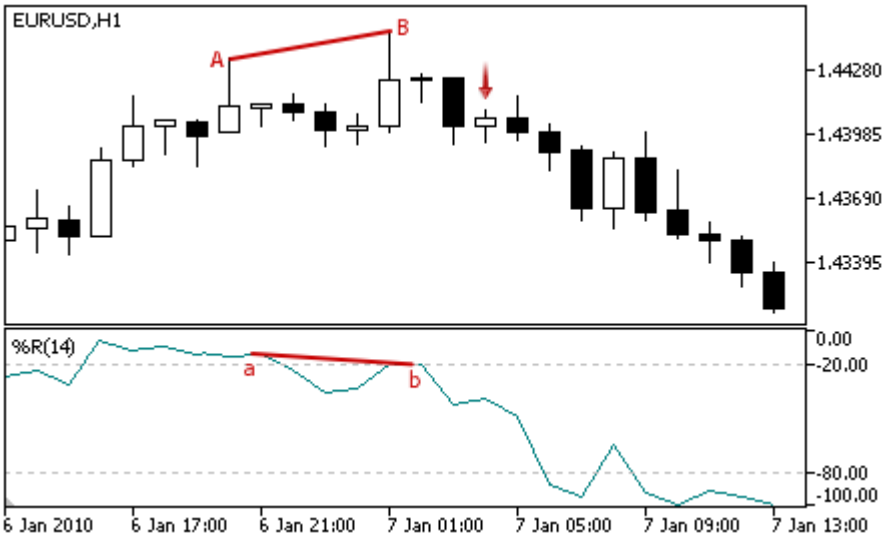
Oscillator Williams Percent Range Osilatörünün Sinyalleri

Bu sinyal modülü, Williams Percent Range osilatörünün piyasa modellerini temel alır. Modüllerden elde edilen sinyallere dayanan alım-satım karar mekanizmaları ayrı bir bölümde ele alınmıştır.

Sinyallerin Oluşma koşulları

Aşağıda, modülün Uzman Danışmana sinyal gönderme koşullarıyla ilgili açıklamalar bulabilirsiniz.

Sinyal Tipi	Koşulların Açıklaması
Alış için	• **Aşırı-satış seviyesinin ardından ters dönüş** — osilatör yukarı yönelmiş ve incelenen mevcut çubuk üzerindeki değeri aşırı-satış seviyesini geçmiş (varsayılan değer : 20).  • **Ayrışma** — osilatörün analiz edilen ilk tabanı bir öncekinden daha yüksek, ve karşılık gelen fiyat tabanı bir öncekinden daha düşük. 
Satış için	• **Aşırı-alış seviyesinden dönüş** — osilatör aşağı yönelmiş ve incelenen mevcut çubuk üzerindeki değeri aşırı-alış seviyesini geçmiş (varsayılan

Sinyal Tipi	Koşulların Açıklaması
	değer : 80).  • **Ayrışma** — osilatörün analiz edilen ilk zirvesi bir öncekinden daha düşük, ve karşılık gelen fiyat zirvesi bir öncekinden daha yüksek. 
Alışa itiraz yok	Osilatör değeri incelenen çubuk üzerinde artıyor.
Satışa itiraz yok	Osilatör değeri incelenen çubuk üzerinde düşüyor.

Not

Uzman Danışmanın işlem kipine bağlı olarak ("Her Tik" veya "Sadece Açılış fiyatları") incelenen çubuk ya mevcut çubuktur (0 indisli) ya da son şekillenen çubuktur (1 indisli).

Williams Percent Range osilatörünün ters ölçeğe sahip olduğunu hatırlayın. Maksimum değeri -100, minimum değeri ise 0.

Ayarlanabilen Parametreler

Bu modül aşağıda belirtilen ayarlanabilir parametreler sahiptir:

Parameter	Açıklama
Weight	0-1 aralığında, modül sinyalinin ağırlığı.
PeriodWPR	Osilatörün hesaplanmasında kullanılan periyot değeri.

Trailing Stop Sınıfları

Bu bölüm, trailing (iz-süren) stop sınıfları ile çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için gereken açıklamaları içermektedir.

Bu sınıfların kullanımı alım-satım stratejilerinin oluşturulması ve sınanması sırasında programcıya zaman kazandıracaktır.

MQL5 Standart Kütüphanesinin trailing (iz-sürme) sınıfları terminalin çalışma dizininde Include\Expert\Trailing klasöründe yer almaktadır.

Sınıf	Açıklama
CTrailingFixedPips	Bu sınıf, sabit puanlara dayanan Trailing Stop algoritması uygular
CTrailingMA	Bu sınıf, hareketli ortalama göstergesinin değerlerine dayanan Trailing Stop algoritması uygular
CTrailingNone	Bir saplama (stub) sınıf, herhangi bir Trailing Stop algoritması uygulamaz
CTrailingPSAR	Bu sınıf, Parabolic SAR göstergesinin değerlerine dayanan Trailing Stop algoritması uygular

CTrailingFixedPips

CTrailingFixedPips, sabit puanlara dayanan Trailing (iz-süren) Stop algoritmasının uygulama sınıfıdır.

Pozisyon Stop Loss seviyesine sahipse, izin verilen minimum Stop Loss - mevcut fiyat uzaklığını denetler. Bu değer Stop Loss seviyesinden düşükse yeni bir Stop Loss fiyatı ayarlamak için öneri sunar. Aynı şekilde pozisyon Take Profit fiyatına sahipse, yeni Take Profit ayarlamayı önerir.

Uzman Danışman, every_tick=false bayrağıyla başlatılmışsa tüm işlemleri (alım-satım, trailing, vb.) sadece yeni çubukta gerçekleştirecektir. Bu durumda Take profit seviyesi kullanılabilir. Bu yeni çubuk tamamlanmadan önce açık pozisyonu Take Profit fiyatından kapamanızı sağlar.

Açıklama

CTrailingFixedPips sınıfı, sabit puanlarla taşınan pozisyonları temel alan Trailing (iz-süren) Stop algoritmasını uygular.

Bildirim

```
class CTrailingFixedPips: public CExpertTrailing
```

Başlık

```
   #include <Expert\Trailing\CTrailingFixedPips.mqh>
```

Kalıtım hiyerarşisi

CObject
  CExpertBase
    CExpertTrailing
      CTrailingFixedPips

Sınıf Yöntemleri

Başlatma	
StopLevel	Stop Loss seviyesini ayarlar
ProfitLevel	Take Profit seviyesini ayarlar
virtual ValidationSettings	Ayarları denetler
Trailing Denetim Yöntemleri	
virtual CheckTrailingStopLong	Uzun pozisyon için Trailing Stop (iz-süren stop) koşularını denetler
virtual CheckTrailingStopShort	Kısa pozisyon için Trailing Stop koşularını denetler

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CExpertBase

Sınıftan türetilen yöntemler 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

StopLevel

Stop Loss seviyesinin değerini puan cinsinden ayarlar.

```
void  StopLevel(
   int  stop_level      // Stop Loss seviyesi
   )
```

Parametreler

stop_loss

[in] Stop Loss seviyesinin (2/4-basamaklı olarak puan cinsinden) yeni değer.

Not

Stop Loss seviyesi 0 olarak ayarlanmışsa, Trailing Stop kullanılmaz.

ProfitLevel

Take Profit seviyesini puan cinsinden ayarlar.

```
void  ProfitLevel(
   int  profit_level      // Take Profit seviyesi
   )
```

Parametreler

profit_level

[in] Take Profit seviyesinin (2/4-basamaklı olarak puan cinsinden) yeni değeri.

Not

Kar seviyesi 0 olarak ayarlanmışsa, Trailing Stop kullanılmaz.

ValidationSettings

Ayarları denetler.

```
virtual bool  ValidationSettings()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Fonksiyon Take Profit ve Stop Loss seviyelerini denetler. Geçerli değer 0 'dır ve değerler sembol üzerindeki stop emirleri için minimal stop değerinden büyüktür.

CheckTrailingStopLong

Uzun pozisyon için Trailing Stop (iz-süren stop) koşularını denetler.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // CPositionInfo nesnesinin işaretçisi
   double&         sl,           // Stop Loss fiyatı
   double&         tp            // Take Profit fiyatı
   )
```

Parametreler

position

[in] CPositionInfo nesnesinin işaretçisi.

sl

[in][out] Stop Loss fiyatının değişkeni.

tp

[in][out] Take fiyatının değişkeni.

Dönüş değeri

Koşullar sağlanmışsa 'true', aksi durumda 'false'.

Not

Stop Loss seviyesi 0 olarak ayarlanmışsa, Trailing Stop kullanılmaz. Pozisyon zaten bir Stop Loss fiyatına sahipse, değeri temel fiyat olarak kabul edilir, aksi durumda temel fiyat olarak pozisyon açılış fiyatı alınır.

Mevcut Satış fiyatı "fiyat+zararı durdur seviyesi" değerinden büyükse, yeni bir Stop Loss fiyatı ayarlamayı önerir. Bu durumda, pozisyon zaten bir Take Profit fiyatına sahipse, yeni Take Profit fiyatı olarak "Bid (teklif)fiyatı+kar al seviyesi" değerinin ayarlanmasını önerir.

CheckTrailingStopShort

Kısa pozisyon için Trailing (iz-süren) Stop koşularını denetler

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // CPositionInfo nesnesinin işaretçisi
   double&         sl,           // Stop Loss fiyatı
   double&         tp            // Take Profit fiyatı
   )
```

Parametreler

position

[in] CPositionInfo nesnesinin işaretçisi.

sl

[in][out] Stop Loss fiyatının değişkeni.

tp

[in][out] Take fiyatının değişkeni.

Dönüş değeri

Koşullar sağlanmışsa 'true', aksi durumda 'false'.

Not

Stop Loss seviyesi 0 olarak ayarlanmışsa, Trailing Stop kullanılmaz. Pozisyon zaten bir Stop Loss fiyatına sahipse, değeri temel fiyat olarak kabul edilir, aksi durumda temel fiyat olarak pozisyon açılış fiyatı alınır.

Mevcut istek (Ask) fiyatı "temel fiyat-zararı durdur seviyesi" değerinden küçükse, yeni bir Stop Loss fiyatı ayarlamayı önerir. Bu durumda, pozisyon zaten bir Take Profit fiyatına sahipse, yeni Take Profit fiyatı olarak "Ask (istek) fiyatı-kar al seviyesi" değerinin ayarlanmasını önerir.

CTrailingMA

CTrailingMA sınıfı, hareketli ortalama göstergesinin değerlerine dayanan Trailing Stop algoritmasının uygulamalarını içerir.

Açıklama

CTrailingMA sınıfı, önceki (tamamlanan) çubuktaki hareketli ortalama göstergesi değerlerine dayanan Trailing Stop algoritması uygular.

Bildirim

```
class CTrailingMA: public CExpertTrailing
```

Başlık

```
   #include <Expert\Trailing\TrailingMA.mqh>
```

Kalıtım hiyerarşisi

CObject
  CExpertBase
    CExpertTrailing
      CTrailingMA

Sınıf Yöntemleri

Başlatma	
Period	Hareketli ortalama periyodunu ayarlar
Shift	Hareketli ortalamanın kaydırma değerini ayarlar
Method	Hareketli ortalamanın düzleştirme yöntemini ayarlar
Applied	Hareketli ortalamanın uygulanan fiyatını ayarlar
virtual InitIndicators	Göstergeleri ve zaman serilerini başlatır
virtual ValidationSettings	Ayarları denetler
Trailing Denetim Yöntemleri	
virtual CheckTrailingStopLong	Uzun pozisyon için Trailing Stop (iz-süren stop) koşullarını denetler
virtual CheckTrailingStopShort	Kısa pozisyon için Trailing Stop koşullarını denetler

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CExpertBase

Sınıftan türetilen yöntemler 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

Hareketli ortalama periyodunu ayarlar.

```
void  Period(
   int  period      // Düzeltme periyodu
   )
```

Parametreler

period

[in] Hareketli ortalama periyodu.

Shift

Hareketli ortalamanın kaydırma değerini ayarlar.

```
void  Shift(
   int  shift      // Kaydırma değeri
   )
```

Parametreler

shift

[in] Hareketli ortalamanın kaydırma değeri.

Method

Hareketli ortalamanın düzleştirme yöntemini ayarlar.

```
void  Method(
   ENUM_MA_METHOD  method      // Düzgünleştirme yöntemi
   )
```

Parametreler

method

[in] Hareketli ortalamanın düzleştirme yöntemi.

Applied

Hareketli ortalamanın uygulanan fiyatını ayarlar.

```
void  Applied(
   ENUM_APPLIED_PRICE  applied      // Uygulanan fiyat
   )
```

Parametreler

applied

[in] Hareketli ortalamanın uygulanan fiyatı.

InitIndicators

Göstergeleri ve zaman serilerini başlatır.

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // CIndicators koleksiyon işaretçisi
   )
```

Parametreler

indicators

[in] Gösterge ve zaman serileri koleksiyonunun işaretçisi (CExpert class member).

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

ValidationSettings

Ayarları denetler.

```
virtual bool  ValidationSettings()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Fonksiyon, hareketli ortalamanın periyot değerini kontrol eder, geçerli değerler pozitiftir.

CheckTrailingStopLong

Uzun pozisyon için Trailing Stop (iz-süren stop) koşularını denetler.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // CPositionInfo nesnesinin işaretçisi
   double&         sl,           // Stop Loss fiyatı
   double&         tp            // Take Profit fiyatı
   )
```

Parametreler

position

[in] CPositionInfo nesnesinin işaretçisi.

sl

[in][out] Stop Loss fiyatının değişkeni.

tp

[in][out] Take fiyatının değişkeni.

Dönüş değeri

Koşullar sağlanmışsa 'true', aksi durumda 'false'.

Not

Önce izin verilen ve mevcut fiyata en yakın olan maksimum Stop Loss fiyatını hesaplar, sonra, bir önceki çubuktaki hareketli ortalama değerlerini kullanarak Stop Loss hesaplar.

Pozisyon zaten bir Stop Loss fiyatına sahipse, değeri temel fiyat olarak kabul edilir, aksi durumda temel fiyat olarak pozisyon açılış fiyatı alınır.

Hesaplanan Stop Loss fiyatı temel fiyattan büyükse ve izin verilen maksimum Stop Loss fiyatından küçükse, yeni Stop Loss fiyatı ayarlamayı önerir.

CheckTrailingStopShort

Kısa pozisyon için Trailing (iz-süren) Stop koşularını denetler

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // CPositionInfo nesnesinin işaretçisi
   double&         sl,           // Stop Loss fiyatı
   double&         tp            // Take Profit fiyatı
   )
```

Parametreler

position

[in] CPositionInfo nesnesinin işaretçisi.

sl

[in][out] Stop Loss fiyatının değişkeni.

tp

[in][out] Take fiyatının değişkeni.

Dönüş değeri

Koşullar sağlanmışsa 'true', aksi durumda 'false'.

Not

Önce izin verilen ve mevcut fiyata en yakın olan minimum Stop Loss fiyatını hesaplar, sonra, bir önceki çubuktaki hareketli ortalama değerlerini kullanarak Stop Loss hesaplar.

Pozisyon zaten bir Stop Loss fiyatına sahipse, değeri temel fiyat olarak kabul edilir, aksi durumda temel fiyat olarak pozisyon açılış fiyatı alınır.

Hesaplanan Stop Loss fiyatı temel fiyattan büyükse ve izin verilen minimum Stop Loss fiyatından küçükse, yeni Stop Loss fiyatı ayarlamayı önerir.

CTrailingNone

CTrailingNone bir "saplama" sınıftır. Bu sınıf, stratejinizde Tailing Stop kullanmıyorsanız, Trailing nesnesinin başlatılması sırasında kullanılmalıdır.

Açıklama

CTrailingNone sınıfı herhangi bir Trailing Stop algoritması uygulamaz. Trailing Stop koşullarının denetim yöntemleri daima 'false' dönüşü yapar.

Bildirim

```
class CTrailingNone: public CExpertTrailing
```

Başlık

```
   #include <Expert\Trailing\TrailingNone.mqh>
```

Kalıtım hiyerarşisi

CObject
  CExpertBase
    CExpertTrailing
      CTrailingNone

Sınıf Yöntemleri

Trailing Denetim Yöntemleri	
virtual CheckTrailingStopLong	Uzun pozisyon için Trailing Stop (iz-süren stop) koşularını denetlemek için bir saplama yöntem
virtual CheckTrailingStopShort	Kısa pozisyon için Trailing Stop koşularını denetlemek için bir saplama yöntem

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries, InitIndicators

Sınıftan türetilen yöntemler CExpertTrailing

CheckTrailingStopLong, CheckTrailingStopShort

CheckTrailingStopLong

Uzun pozisyon için Trailing Stop (iz-süren stop) koşularını denetler.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,     // CPositionInfo nesnesinin işaretçisi
   double&         sl,           // Stop Loss fiyatı
   double&         tp            // Take Profit fiyatı
   )
```

Parametreler

position

[in] CPositionInfo nesnesinin işaretçisi.

sl

[in][out] Stop Loss fiyatının değişkeni.

tp

[in][out] Take fiyatının değişkeni.

Dönüş değeri

Koşullar sağlanmışsa 'true', aksi durumda 'false'.

Not

Fonksiyon her zaman 'false' dönüşü yapar.

CheckTrailingStopShort

Kısa pozisyon için Trailing (iz-süren) Stop koşularını denetler

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // CPositionInfo nesnesinin işaretçisi
   double&         sl,           // Stop Loss fiyatı
   double&         tp            // Take Profit fiyatı
   )
```

Parametreler

position

[in] CPositionInfo nesnesinin işaretçisi.

sl

[in][out] Stop Loss fiyatının değişkeni.

tp

[in][out] Take fiyatının değişkeni.

Dönüş değeri

Koşullar sağlanmışsa 'true', aksi durumda 'false'.

Not

Fonksiyon her zaman 'false' dönüşü yapar.

CTrailingPSAR

CTrailingPSAR sınıfı, Parabolic SAR göstergesinin değerlerine dayanan Trailing Stop (iz-süren stop) algoritmasının uygulamalarını içerir.

Açıklama

CTrailingPSAR sınıfı, önceki (tamamlanan) çubuktaki Parabolic SAR göstergesi değerlerine dayanan Trailing Stop algoritması uygular.

Bildirim

```
class CTrailingPSAR: public CExpertTrailing
```

Başlık

```
   #include <Expert\Trailing\TrailingParabolicSAR.mqh>
```

Kalıtım hiyerarşisi

CObject
  CExpertBase
    CExpertTrailing
      CTrailingPSAR

Sınıf Yöntemleri

Başlatma	
Step	Parabolic SAR göstergesinin adım değerini ayarlar
Maximum	Parabolic SAR göstergesinin maksimum değerini ayarlar
virtual InitIndicators	Göstergeleri ve zaman serilerini başlatır
Trailing Denetim Yöntemleri	
virtual CheckTrailingStopLong	Uzun pozisyon için Trailing Stop (iz-süren stop) koşullarını denetler
virtual CheckTrailingStopShort	Kısa pozisyon için Trailing Stop koşullarını denetler

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, ValidationSettings, SetPriceSeries, SetOtherSeries

Step

Parabolic SAR göstergesinin adım değerini ayarlar.

```
void  Step(
   double  step      // Adım
   )
```

Parametreler

step

[in] Parabolic SAR göstergesinin adım değeri.

Maximum

Parabolic SAR göstergesinin maksimum değerini ayarlar.

```
void  Maximum(
   double  maximum      // Maksimum
   )
```

Parametreler

maximum

[in] Parabolic SAR göstergesinin maksimum değeri.

InitIndicators

Göstergeleri ve zaman serilerini başlatır.

```
virtual bool  InitIndicators(
   CIndicators*  indicators      // CIndicators koleksiyon işaretçisi
   )
```

Parametreler

indicators

[in] Gösterge ve zaman serileri koleksiyonunun işaretçisi (CExpert class member).

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CheckTrailingStopLong

Uzun pozisyon için Trailing Stop (iz-süren stop) koşularını denetler.

```
virtual bool  CheckTrailingStopLong(
   CPositionInfo*  position,      // işaretçi
   double&         sl,            // Stop Loss seviyesi
   double&         tp             // Take Profit seviyesi
   )
```

Parametreler

position

[in] CPositionInfo nesnesinin işaretçisi.

sl

[in][out] Stop Loss fiyatının değişkeni.

tp

[in][out] Take fiyatının değişkeni.

Dönüş değeri

Koşullar sağlanmışsa 'true', aksi durumda 'false'.

Not

Önce izin verilen ve mevcut fiyata en yakın olan maksimum Stop Loss fiyatını hesaplar, sonra, bir önceki çubuktaki Parabolic SAR değerlerini kullanarak Stop Loss hesaplar.

Pozisyon zaten bir Stop Loss fiyatına sahipse, değeri temel fiyat olarak kabul edilir, aksi durumda temel fiyat olarak pozisyon açılış fiyatı alınır.

Hesaplanan Stop Loss fiyatı temel fiyattan büyükse ve izin verilen maksimum Stop Loss fiyatından küçükse, yeni Stop Loss fiyatı ayarlamayı önerir.

CheckTrailingStopShort

Kısa pozisyon için Trailing (iz-süren) Stop koşularını denetler

```
virtual bool  CheckTrailingStopShort(
   CPositionInfo*  position,     // işaretçi
   double&         sl,           // Stop Loss seviyesi
   double&         tp            // Take Profit seviyesi
   )
```

Parametreler

position

[in] CPositionInfo nesnesinin işaretçisi.

sl

[in][out] Stop Loss fiyatının değişkeni.

tp

[in][out] Take fiyatının değişkeni.

Dönüş değeri

Koşullar sağlanmışsa 'true', aksi durumda 'false'.

Not

Önce izin verilen ve mevcut fiyata en yakın olan minimum Stop Loss fiyatını hesaplar, sonra, bir önceki çubuktaki Parabolic SAR değerlerini kullanarak Stop Loss hesaplar.

Pozisyon zaten bir Stop Loss fiyatına sahipse, değeri temel fiyat olarak kabul edilir, aksi durumda temel fiyat olarak pozisyon açılış fiyatı alınır.

Hesaplanan Stop Loss fiyatı temel fiyattan büyükse ve izin verilen minimum Stop Loss fiyatından küçükse, yeni Stop Loss fiyatı ayarlamayı önerir.

Para Yönetimi Sınıfları

Bu bölüm, para ve risk yönetimi sınıflarıyla çalışmanın teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için gereken açıklamaları içermektedir.

Bu sınıfların kullanımı alım-satım stratejilerinin oluşturulması ve sınanması sırasında programcıya zaman kazandıracaktır.

MQL5 Standart Kütüphanesinin para ve risk yönetimi sınıfları, terminalin çalışma dizininde Include\Expert\Money\ klasöründe yer almaktadır.

Sınıf	Açıklama
CMoneyFixedLot	Bu sınıf, önceden belirlenmiş sabit lot değerini temel alan para yönetimi algoritmasını uygular.
CMoneyFixedMargin	Bu sınıf, önceden belirlenmiş sabit teminat değerini temel alan para yönetimi algoritmasını uygular.
CMoneyFixedRisk	Bu sınıf, önceden belirlenmiş risk değerini temel alan para yönetimi algoritmasını uygular.
CMoneyNone	Bu sınıf, kullanılabilir en küçük lot değerini temel alan para yönetimi algoritmasını uygular.
CMoneySizeOptimized	Bu sınıf, önceki işlemlerin sonuçlarına göre belirlenen değişken lot değeri ile işlem yapmak için düzenlenmiş para yönetimi algoritmasını uygular.

CMoneyFixedLot

CMoneyFixedLot sınıfı, önceden belirlenmiş sabit lot değeri ile işlem yapmak için düzenlenmiş bir para yönetimi sınıfıdır.

Açıklama

CMoneyFixedLot önceden belirlenmiş sabit lot değerini temel alan para yönetimi algoritmasını uygular.

Bildirim

```
class CMoneyFixedLot: public CExpertMoney
```

Başlık

```
#include <Expert\Money\MoneyFixedLot.mqh>
```

Kalıtım hiyerarşisi

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedLot

Sınıf Yöntemleri

Başlatma	
Lots	İşlem hacmini ayarlar
virtual ValidationSettings	Ayarları denetler
Para ve Risk Yönetimi Yöntemleri	
virtual CheckOpenLong	Uzun pozisyon için işlem hacmini alır
virtual CheckOpenShort	Kısa pozisyon için işlem hacmini alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Sınıftan türetilen yöntemler CExpertMoney

Percent, CheckReverse, CheckClose

Lots

İşlem hacmini lot bazında ayarlar

```
void  Lots(
   double  lots      // Lot değeri
   )
```

Parametreler

lots

[in] Lot bazında işlem hacmi.

ValidationSettings

Ayarları denetler.

```
virtual bool  ValidationSettings()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Doğrulama amacıyla, belirtilen işlem hacmi değerini denetler.

CheckOpenLong

Uzun pozisyon için işlem hacmini alır.

```
virtual double  CheckOpenLong(
   double  price,     // fiyat
   double  sl         // Stop Loss fiyatı
   )
```

Parametreler

price

[in] Fiyat.

sl

[in] Stop Loss (zararı durdur) fiyatı.

Dönüş değeri

Uzun pozisyon için işlem hacmi değeri.

Not

Bu fonksiyon her zaman Lots yöntemiyle tanımlanmış sabit işlem hacmi değerine dönüş yapar.

CheckOpenShort

Kısa pozisyon için işlem hacmi değerini alır.

```
virtual double  CheckOpenShort(
   double  price,     // fiyat
   double  sl         // Stop Loss fiyatı
   )
```

Parametreler

price

[in] Fiyat.

sl

[in] Stop Loss (zararı durdur) fiyatı.

Dönüş değeri

Kısa pozisyon için işlem hacmi değeri.

Not

Bu fonksiyon her zaman Lots yöntemiyle tanımlanmış sabit işlem hacmi değerine dönüş yapar.

CMoneyFixedMargin

CMoneyFixedMargin, önceden belirlenmiş sabit teminat değeri ile işlem yapmak için düzenlenmiş bir sınıftır.

Açıklama

CMoneyFixedMargin, sabit teminat ile işlem şeklini temel alan para yönetimi uygulamasıdır.

Bildirim

```
class CMoneyFixedMargin: public CExpertMoney
```

Başlık

```
#include <Expert\Money\MoneyFixedMargin.mqh>
```

Kalıtım hiyerarşisi

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedMargin

Sınıf Yöntemleri

Para ve Risk Yönetimi Yöntemleri	
virtual CheckOpenLong	Uzun pozisyon için işlem hacmini alır
virtual CheckOpenShort	Kısa pozisyon için işlem hacmini alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Sınıftan türetilen yöntemler CExpertMoney

Percent, ValidationSettings, CheckReverse, CheckClose

CheckOpenLong

Uzun pozisyon için işlem hacmini alır.

```
virtual double  CheckOpenLong(
   double  price,     // fiyat
   double  sl         // Stop Loss fiyatı
   )
```

Parametreler

price

[in] Fiyat.

sl

[in] Stop Loss (zararı durdur) fiyatı.

Dönüş değeri

Uzun pozisyon için işlem hacmi değeri.

Not

Fonksiyon, sabit teminat kullanarak uzun pozisyon için kullanılacak işlem hacmine dönüş yapar. Teminat değeri, CExpertMoney temel sınıfının "Percent" (yüzde) parametresi ile belirlenir.

CheckOpenShort

Kısa pozisyon için işlem hacmi değerini alır.

```
virtual double  CheckOpenShort(
   double  price,     // fiyat
   double  sl         // Stop Loss fiyatı
   )
```

Parametreler

price

[in] Fiyat.

sl

[in] Stop Loss (zararı durdur) fiyatı.

Dönüş değeri

Kısa pozisyon için işlem hacmi değeri.

Not

Fonksiyon, sabit teminat kullanarak kısa pozisyon için kullanılacak işlem hacmine dönüş yapar. Teminat değeri, CExpertMoney temel sınıfının "Percent" (yüzde) parametresi ile belirlenir.

CMoneyFixedRisk

CMoneyFixedRisk, önceden belirlenmiş sabit risk değeri ile işlem yapmak için düzenlenmiş bir sınıftır.

Açıklama

CMoneyFixedRisk sınıfı, önceden belirlenmiş bir sabit risk değeri üzerine kurulu bir para yönetimi algoritmasını uygular.

Bildirim

```
class CMoneyFixedRisk: public CExpertMoney
```

Başlık

```
#include <Expert\Money\MoneyFixedRisk.mqh>
```

Kalıtım hiyerarşisi

CObject
  CExpertBase
    CExpertMoney
      CMoneyFixedRisk

Sınıf Yöntemleri

Para ve Risk Yönetimi Yöntemleri	
virtual CheckOpenLong	Uzun pozisyon için işlem hacmini alır
virtual CheckOpenShort	Kısa pozisyon için işlem hacmini alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Sınıftan türetilen yöntemler CExpertMoney

Percent, ValidationSettings, CheckReverse

CheckOpenLong

Uzun pozisyon için işlem hacmini alır.

```
virtual double  CheckOpenLong(
   double  price,     // fiyat
   double  sl         // Stop Loss fiyatı
   )
```

Parametreler

price

[in] Fiyat.

sl

[in] Stop Loss (zararı durdur) fiyatı.

Dönüş değeri

Uzun pozisyon için işlem hacmi değeri.

Not

Fonksiyon, sabit risk kullanarak uzun pozisyon için kullanılacak işlem hacmine dönüş yapar. Risk değeri CExpertMoney temel sınıfının "Percent" (yüzde) parametresi ile belirlenir.

CheckOpenShort

Kısa pozisyon için işlem hacmi değerini alır.

```
virtual double  CheckOpenShort(
   double  price,     // fiyat
   double  sl         // Stop Loss fiyatı
   )
```

Parametreler

price

[in] Fiyat.

sl

[in] Stop Loss (zararı durdur) fiyatı.

Dönüş değeri

Kısa pozisyon için işlem hacmi değeri.

Not

Fonksiyon, sabit risk değeri kullanarak kısa pozisyon için kullanılacak işlem hacmine dönüş yapar. Risk değeri CExpertMoney temel sınıfının "Percent" (yüzde) parametresi ile belirlenir.

CMoneyNone

CMoneyNone, izin verilen en küçük lot değeri ile işlem yapmak için düzenlenmiş bir sınıftır.

Açıklama

CMoneyNone sınıfı, izin verilen en küçük lot değeri ile işlem yapmak için düzenlenmiş para yönetimi algoritmasını uygular.

Bildirim

```
class CMoneyNone: public CExpertMoney
```

Başlık

```
#include <Expert\Money\MoneyNone.mqh>
```

Kalıtım hiyerarşisi

CObject
  CExpertBase
    CExpertMoney
      CMoneyNone

Sınıf Yöntemleri

Başlatma	
virtual ValidationSettings	Ayarları denetler
Para ve Risk Yönetimi Yöntemleri	
virtual CheckOpenLong	Uzun pozisyon için işlem hacmini alır
virtual CheckOpenShort	Kısa pozisyon için işlem hacmini alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Sınıftan türetilen yöntemler CExpertMoney

Percent, CheckReverse, CheckClose

ValidationSettings

Ayarları denetler.

```
virtual bool  ValidationSettings()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Fonksiyon her zaman 'true' dönüşü yapar.

CheckOpenLong

Uzun pozisyon için işlem hacmini alır.

```
virtual double  CheckOpenLong(
   double  price,     // fiyat
   double  sl         // Stop Loss fiyatı
   )
```

Parametreler

price

[in] Fiyat.

sl

[in] Stop Loss (zararı durdur) fiyatı.

Dönüş değeri

Uzun pozisyon için işlem hacmi değeri.

Not

Fonksiyon her zaman kullanılabilecek en küçük lot değerine dönüş yapar.

CheckOpenShort

Uzun pozisyon için işlem hacmini alır.

```
virtual double  CheckOpenShort(
   double  price,     // fiyat
   double  sl         // Stop Loss fiyatı
   )
```

Parametreler

price

[in] Fiyat.

sl

[in] Stop Loss (zararı durdur) fiyatı.

Dönüş değeri

Kısa pozisyon için işlem hacmi değeri.

Not

Fonksiyon her zaman kullanılabilecek en küçük lot değerine dönüş yapar.

CMoneySizeOptimized

CMoneySizeOptimized, önceki işlemlerin sonuçlarına göre belirlenen değişken lot değeri ile işlem yapmak için düzenlenmiş bir para yönetimi sınıfıdır.

Açıklama

CMoneySizeOptimized, önceki işlemlerin sonuçlarına göre belirlenen değişken lot değeri ile işlem yapmak için düzenlenmiş para yönetimi algoritmasını uygular.

Bildirim

```
class CMoneySizeOptimized: public CExpertMoney
```

Başlık

```
#include <Expert\Money\MoneySizeOptimized.mqh>
```

Kalıtım hiyerarşisi

CObject
  CExpertBase
    CExpertMoney
      CMoneySizeOptimized

Sınıf Yöntemleri

Başlatma	
DecreaseFactor	Azaltma faktörünün değerini ayarlar
virtual ValidationSettings	Ayarları denetler
Para ve Risk Yönetimi Yöntemleri	
virtual CheckOpenLong	Uzun pozisyon için işlem hacmini alır
virtual CheckOpenShort	Kısa pozisyon için işlem hacmini alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler CExpertBase

InitPhase, TrendType, UsedSeries, EveryTick, Open, High, Low, Close, Spread, Time, TickVolume, RealVolume, Init, Symbol, Period, Magic, SetMarginMode, SetPriceSeries, SetOtherSeries, InitIndicators

Sınıftan türetilen yöntemler CExpertMoney

Percent, CheckReverse, CheckClose

DecreaseFactor

Azaltma faktörünün değerini ayarlar.

```
void  DecreaseFactor(
   double  decrease_factor      // azaltma faktörü değeri
   )
```

Parametreler

decrease_factor

[in] Azaltma faktörü değeri.

Not

DecreaseFactor fonksiyonu, ardışık kayıp durumlarında önceki pozisyonun hacmiyle kıyaslayarak hacim azaltma katsayısını tanımlar.

ValidationSettings

Ayarları denetler.

```
virtual bool  ValidationSettings()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Azaltma faktörünün değeri negatif ise 'false' değerine, aksi durumda 'true' değerine dönüş yapar.

CheckOpenLong

Uzun pozisyon için işlem hacmini alır.

```
virtual double  CheckOpenLong(
   double  price,     // fiyat
   double  sl         // Stop Loss fiyatı
   )
```

Parametreler

price

[in] Fiyat.

sl

[in] Stop Loss (zararı durdur) fiyatı.

Dönüş değeri

Uzun pozisyon için işlem hacmi değeri.

Not

Fonksiyon, uzun pozisyon için kullanılacak işlem hacmi değerine dönüş yapar, hacim değeri önceki işlemlerin sonucuna bağlıdır.

CheckOpenShort

Kısa pozisyon için işlem hacmi değerini alır.

```
virtual double  CheckOpenShort(
   double  price,     // fiyat
   double  sl         // Stop Loss fiyatı
   )
```

Parametreler

price

[in] Fiyat.

sl

[in] Stop Loss (zararı durdur) fiyatı.

Dönüş değeri

Uzun pozisyon için işlem hacmi değeri.

Not

Fonksiyon, kısa pozisyon için kullanılacak işlem hacmi değerine dönüş yapar, hacim değeri önceki işlemlerin sonucuna bağlıdır.

Kontrol Panelleri ve Diyaloglar Oluşturmak için Sınıflar

Bu bölüm, kontrol panelleri oluşturmak için tasarlanmış sınıfların teknik detaylarını ve MQL5 Standart Kütüphanesinin ilgili kısımları için gereken açıklamaları içermektedir.

Bu sınıfların kullanımı, MQL5 programları (Uzman Danışmanlar ve Göstergeler) için kontrol panelleri oluştururken zaman kazandıracaktır.

MQL5 Standart kütüphanesinin kontrollerle ilgili sınıfları müşteri terminalinin veri klasöründe, MQL5\Include\Controls dizininde yer almaktadır.

Sınıflarla çalışma örnekleri aşağıdaki makalelerde bulunabilir:

- Herhangi bir karmaşıklık düzeyinde bir grafik paneli nasıl oluşturulur?
- Panelleri Geliştirme: Saydamlık ekleme, arka plan rengini değiştirme ve CAppDialog/CWndClient'tan devralma
- Bir göstergeye veya bir Uzman Danışmana hızlı bir şekilde kontrol paneli ekleme
- MQL5'te kendi grafik panellerinizi oluşturma
- Ticaret için MQL5'te aktif kontrol panelleri oluşturma

Bu sınıflarla yapılan örnek bir çalışmaya (bir Uzman Danışman örneği) MQL5\Expert\Examples\Controls dizininden ulaşılabilir.

Yardımcı yapılar	Açıklama
CRect	Dikdörtgen alan yapısı
CDateTime	Tarih ve zaman ile çalışmak için bir yapı

Temel sınıflar	Açıklama
CWnd	Tüm kontroller için temel sınıf
CWndObj	Kontroller ve iletişim panelleri için temel sınıf
CWndContainer	Karmaşık (bağımlı kontroller içeren) kontroller için temel yapı

Basit kontroller	Açıklama
CLabel	"Metin etiketi" grafik nesnesini temel alan kontrol
CBmpButton	"Biteşlem etiketi" grafik nesnesini temel alan kontrol
CButton	"Düğme" grafik nesnesini temel alan kontrol
CEdit	"Düzenleme alanı" grafik nesnesini temel alan kontrol
CPanel	"Dikdörtgen etiket" grafik nesnesini temel alan kontrol

Basit kontroller	Açıklama
CPicture	"Biteşlem etiketi" grafik nesnesini temel alan kontrol

Karmaşık kontroller	Açıklama
CScroll	Kaydırma çubuğu için temel sınıf
CScrollV	Dikey kaydırma çubuğu
CScrollH	Yatay kaydırma çubuğu
CWndClient	Kaydırma çubuklu müşteri alanı için temel sınıf
CListView	ListView
CComboBox	ComboBox
CCheckBox	CheckBox
CCheckGroup	CheckGroup
CRadioButton	RadioButton
CRadioGroup	RadioGroup
CSpinEdit	SpinEdit
CDialog	Diyalog
CAppDialog	Uygulama Diyaloğu

CRect

CRect bir dikdörtgen çizelge alanı sınıfıdır.

Açıklama

CRect bir alan sınıfıdır, dikdörtgenin sol-üst ve sağ-alt köşelerinin kartezyen koordinatlarıyla tanımlanır.

Bildirim

```
class CRect
```

Başlık

```
#include <Controls\Rect.mqh>
```

Sınıf Yöntemleri

Özellikler	
Left	Sol-üst köşenin X koordinatını alır/ayarlar
Top	Sol-üst köşenin Y koordinatını alır/ayarlar
Right	Sağ-alt köşenin X koordinatını alır/ayarlar
Bottom	Sağ-alt köşenin Y koordinatını alır/ayarlar
Width	Genişlik değerini alır/ayarlar
Height	Yükseklik değerini alır/ayarlar
SetBound	CRect dikdörtgen alanını için yeni koordinatlar ayarlar
Move	CRect dikdörtgen alanının hareketi için yeni koordinatlar ayarlar
Shift	Crect dikdörtgen alanını göreli olarak hareket ettirir
Contains	Noktanın CRect dikdörtgen alanı içinde olup olmadığını kontrol eder
Ek yöntemler	
Format	Alan koordinatlarını dizgi biçiminde alır

Left (Get Yöntemi)

Sol-üst köşenin X koordinatını alır.

```
int  Left()
```

Dönüş değeri

Sol-üst köşenin X koordinatı.

Left (Set Yöntemi)

Sol-üst köşenin X koordinatını ayarlar.

```
void  Left(
   const int  x      // yeni x koordinatı
   )
```

Parametreler

x

[in] Sol-üst köşenin yeni X koordinatı.

Dönüş değeri

Yok.

Top (Get Yöntemi)

Sol-üst köşenin Y koordinatını alır.

```
int  Top()
```

Dönüş değeri

Sol-üst köşenin Y koordinatı.

Top (Set Yöntemi)

Sol-üst köşenin Y koordinatını ayarlar.

```
void  Top(
   const int  y      // y koordinatı
   )
```

Parametreler

y

[in] Sol-üst köşenin Y koordinatı için yeni değer.

Dönüş değeri

Yok.

Right (Get Yöntemi)

Sağ-alt köşenin X koordinatını alır.

```
int  Right()
```

Dönüş değeri

Sağ-alt köşenin X koordinatı.

Right (Set Yöntemi)

Sağ-alt köşenin Y koordinatını ayarlar.

```
void  Right(
   const int  x      // x koordinatı
   )
```

Parametreler

x

[in] Sağ-alt köşe için yeni X koordinatı.

Dönüş değeri

Yok.

Bottom (Get Yöntemi)

Sağ-alt köşenin Y koordinatını alır.

```
int  Bottom()
```

Dönüş değeri

Sağ-alt köşenin Y koordinatı.

Bottom (Set Yöntemi)

Sağ-alt köşenin Y koordinatını ayarlar.

```
void  Bottom(
   const int  y      // y koordinatı
   )
```

Parametreler

y

[in] Sağ-alt köşenin Y koordinatı için yeni değer.

Dönüş değeri

Yok.

Width (Get Yöntemi)

Alan genişliğini alır.

```
int  Width()
```

Dönüş değeri

Alan genişliği.

Width (Set Yöntemi)

Alan genişliğini ayarlar.

```
virtual bool  Width(
   const int  w      // genişlik
   )
```

Parametreler

w

[in] Yeni genişlik değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Height (Get Yöntemi)

Alan yüksekliğini alır.

```
int  Height()
```

Dönüş değeri

Alan yüksekliği.

Height (Set Yöntemi)

Alan yüksekliğini ayarlar.

```
virtual bool  Height(
   const int  h      // yükseklik
   )
```

Parametreler

h

[in] Yeni yükseklik.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

SetBound

CRect sınıf koordinatlarını kullanarak yeni koordinatlar ayarlar.

```
void  SetBound(
   const & CRect  rect      // CRect sınıfı
   )
```

Dönüş değeri

Yok.

SetBound

Alan için yeni koordinatlar ayarlar.

```
void  SetBound(
   const int  l      // sol
   const int  t      // üst
   const int  r      // sağ
   const int  b      // alt
   )
```

Parametreler

l

[in] Sol üst köşenin X koordinatı.

t

[in] Sol üst köşenin Y koordinatı.

r

[in] Sağ alt köşenin X koordinatı.

b

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Yok.

Move

CRect dikdörtgen alanı için kesin konum koordinatları oluşturur.

```
void  Move(
   const int  x,      // X koordinatı
   const int  y       // Y koordinatı
   )
```

Parametreler

x

[in] Yeni X koordinatı.

y

[in] Yeni Y koordinatı.

Dönüş değeri

Yok.

Shift

Crect dikdörtgen alanını göreli olarak hareket ettirir.

```
void  Shift(
   const int  dx,      // delta X
   const int  dy       // delta Y
   )
```

Parametreler

dx

[in] Delta X.

dy

[in] Delta Y.

Dönüş değeri

Yok.

Contains

Checks if the point is inside the CRect class area.

```
bool  Contains(
   const int  x,      // X koordinatı
   const int  y       // Y koordinatı
   )
```

Parametreler

x

[in] X koordinatı.

y

[in] Y koordinatı.

Dönüş değeri

Nokta, alanın içindeyse 'true', aksi durumda 'false'..

Format

Alan koordinatlarını dizgi biçiminde alır.

```
string  Format(
   string &  fmt,     // format
   ) const
```

Parametreler

fmt

[in] Formatlı dizgi.

Dönüş değeri

Dizgi biçiminde alan koordinatları.

CDateTime

CDateTime, tarih ve zaman ile çalışmak için tasarlanmış bir yapıdır.

Açıklama

CDateTime yapısı, kontroller üzerinde tarih ve zaman işlemleri yapmak için MqlDateTime yapısından türetilmiştir.

Bildirim

```
struct CDateTime
```

Başlık

```
#include <Tools\DateTime.mqh>
```

Sınıf Yöntemleri

Özellikler	
MonthName	Ay ismini alır
ShortMonthName	Ayın kısa ismini alır
DayName	Hafta gününün ismini alır
ShortDayName	Hafta gününün kısa ismini alır
DaysInMonth	Ay içindeki gün sayısını alır
Get/Set yöntemleri	
DateTime	Tarih ve zaman bilgisini alır/ayarlar
Date	Tarihi ayarlar
Time	Zamanı ayarlar
Sec	Saniye değerini ayarlar
Min	Dakika değerini ayarlar
Hour	Saat değerini ayarlar
Day	Gün değerini ayarlar
Mon	Ay değerini ayarlar
Year	Yıl değerini ayarlar
Ek yöntemler	
SecDec	Belirtilen sayıda saniye çıkarır
SecInc	Belirtilen sayıda saniye ekler
MinDec	Belirtilen sayıda dakika çıkarır

Özellikler	
MinInc	Belirtilen sayıda dakika ekler
HourDec	Belirtilen sayıda saat çıkarır
HourInc	Belirtilen sayıda saat ekler
DayDec	Belirtilen sayıda gün çıkarır
DayInc	Belirtilen sayıda gün ekler
MonDec	Belirtilen sayıda ay çıkarır
MonInc	Belirtilen sayıda ay ekler
YearDec	Belirtilen sayıda yıl çıkarır
YearInc	Belirtilen sayıda yıl ekler

MonthName

Ay ismini alır.

```
string  MonthName()  const
```

İndis numarasına göre ay ismini alır.

```
string  MonthName(
   const int       num          // ay ismi
   )  const
```

Parametreler

num

[in] Ay indisi (1-12).

Dönüş değeri

Ay ismi.

ShortMonthName

Ayın kısa ismini alır.

```
string  ShortMonthName()  const
```

Ayın kısa ismini indis kullanarak alır.

```
string  ShortMonthName(
   const int       num          // ay ismi
   )  const
```

Parametreler

num

[in] Ay indisi (1-12).

Dönüş değeri

Ayın kısa ismi.

DayName

Hafta gününün ismini alır.

```
string  DayName()  const
```

Hafta günü ismini indisine göre alır.

```
string  DayName(
   const int      num         // gün indisi
   )  const
```

Parametreler

num

[in] Gün indisi (0-6).

Dönüş değeri

Günün ismi.

ShortDayName

Hafta gününün kısa ismini alır.

```
string  ShortDayName()  const
```

Hafta gününün kısa ismini indis ile alır.

```
string  ShortDayName(
   const int      num         // gün indisi
   )  const
```

Parametreler

num

[in] Gün indisi (0-6).

Dönüş değeri

Günün kısa ismi.

DaysInMonth

Ay içindeki gün sayısını alır.

```
int  DaysInMonth()  const
```

Dönüş değeri

Ay içindeki günlerin sayısı.

DateTime (Get Yöntemi)

Tarih ve zaman bilgisini alır.

```
datetime  DateTime()
```

Dönüş değeri

datetime tipli bir değer.

DateTime (Set Yöntemi datetime)

Tarih ve zamanı datetime tipi ile ayarlar.

```
void  DateTime(
   const datetime      value         // tarih ve zaman
   )
```

Parametreler

value

[in] datetime tipli değer.

Dönüş değeri

Yok.

DateTime (Set Yöntemi MqlDateTime)

Tarih ve zaman değerlerini MqlDateTime tipi ile ayarlar.

```
void  DateTime(
   const MqlDateTime   &value        // tarih ve zaman
   )
```

Parametreler

value

[in] MqlDateTime tipli değer.

Dönüş değeri

Yok.

Date (Set Yöntemi datetime)

Tarih değerini datetime tipi ile ayarlar.

```
void  Date(
   const datetime      value         // tarih
   )
```

Parametreler

value

[in] datetime tipli değer.

Dönüş değeri

Yok.

Date (Set Yöntemi MqlDateTime)

Tarih değerini MqlDateTime tipi ile ayarlar.

```
void  Date(
   const MqlDateTime   &value        // tarih
   )
```

Parametreler

value

[in] MqlDateTime tipli değer.

Dönüş değeri

Yok.

Time (Set Yöntemi datetime)

Zaman değerini datetime tipi ile ayarlar.

```
void  Time(
   const datetime      value          // zaman
   )
```

Parametreler

value

[in] datetime tipli değer.

Dönüş değeri

Yok.

Time (Set Yöntemi MqlDateTime)

Zaman değerini MqlDateTime tipi ile ayarlar.

```
void  Time(
   const MqlDateTime    &value         // zaman
   )
```

Parametreler

value

[in] MqlDateTime tipli değer.

Dönüş değeri

Yok.

Sec

Saniye değerini ayarlar.

```
void  Sec(
   const int   value        // saniye
   )
```

Parametreler

value

[in] Yeni saniye değeri.

Dönüş değeri

Yok.

Min

Dakika değerini ayarlar.

```
void  Min(
   const int   value        // dakika değeri
   )
```

Parametreler

value

[in] Yeni dakika değeri.

Dönüş değeri

Yok.

Hour

Saat değerini ayarlar.

```
void  Hour(
   const int   value        // saat
   )
```

Parametreler

value

[in] Saat.

Dönüş değeri

Yok.

Day

Ayın günü değerini ayarlar.

```
void  Day(
   const int    value        // gün
   )
```

Parametreler

value

[in] Ayın günü.

Dönüş değeri

Yok.

Mon

Ay değerini ayarlar.

```
void  Mon(
   const int   value        // ay
   )
```

Parametreler

value

[in] Ay değeri.

Dönüş değeri

Yok.

Year

Yıl değerini ayarlar.

```
void  Year(
   const int   value        // yıl
   )
```

Parametreler

value

[in] Yıl.

Dönüş değeri

Yok.

SecDec

Belirtilen sayıda saniye çıkarır.

```
void  SecDec(
   int   delta=1        // saniye sayısı
   )
```

Parametreler

delta

[in] Çıkarılacak saniye sayısı.

Dönüş değeri

Yok.

SecInc

Belirtilen sayıda saniye ekler.

```
void  SecInc(
   int   delta=1        // saniye sayısı
   )
```

Parametreler

delta

[in] Eklenecek saniye sayısı.

Dönüş değeri

Yok.

MinDec

Belirtilen sayıda dakika çıkarır.

```
void  MinDec(
   int   delta=1        // dakika sayısı
   )
```

Parametreler

delta

[in] Çıkarılacak dakika sayısı.

Dönüş değeri

Yok.

MinInc

Belirtilen sayıda dakika ekler.

```
void  MinInc(
   int   delta=1        // dakika sayısı
   )
```

Parametreler

delta

[in] Eklenecek dakika sayısı.

Dönüş değeri

Yok.

HourDec

Belirtilen sayıda saat çıkarır.

```
void  HourDec(
   int   delta=1        // saat
   )
```

Parametreler

delta

[in] Çıkarılacak saat sayısı.

Dönüş değeri

Yok.

HourInc

Belirtilen sayıda saat ekler.

```
void  HourInc(
   int   delta=1        // saat
   )
```

Parametreler

delta

[in] Eklenecek saat.

Dönüş değeri

Yok.

DayDec

Belirtilen sayıda gün çıkarır.

```
void  DayDec(
   int   delta=1        // gün sayısı
   )
```

Parametreler

delta

[in] Çıkarılacak gün sayısı.

Dönüş değeri

Yok.

DayInc

Belirtilen sayıda gün ekler.

```
void  DayInc(
   int   delta=1        // gün sayısı
   )
```

Parametreler

delta

[in] Eklenecek gün sayısı.

Dönüş değeri

Yok.

MonDec

Belirtilen sayıda ay çıkarır.

```
void  MonDec(
   int   delta=1        // ay sayısı
   )
```

Parametreler

delta

[in] Çıkarılacak ay sayısı.

Dönüş değeri

Yok.

MonInc

Belirtilen sayıda ay ekler.

```
void  MonInc(
   int   delta=1        // ay sayısı
   )
```

Parametreler

delta

[in] Eklenecek ay sayısı.

Dönüş değeri

Yok.

YearDec

Belirtilen sayıda yıl çıkarır.

```
void  YearDec(
   int   delta=1        // yıl
   )
```

Parametreler

delta

[in] Çıkarılacak yıl sayısı.

Dönüş değeri

Yok.

YearInc

Belirtilen sayıda yıl ekler.

```
void  YearInc(
   int   delta=1        // yıl
   )
```

Parametreler

delta

[in] Eklenecek yıl sayısı.

Dönüş değeri

Yok.

CWnd

CWnd sınıfı, MQL5 Standart Kütüphanesi içinde yer alan tüm kontroller için bir temel sınıftır.

Açıklama

CWnd sınıfı, temel kontrol sınıfını uygulamasıdır.

Bildirim

```
class CWnd : public CObject
```

Başlık

```
#include <Controls\Wnd.mqh>
```

Kalıtım hiyerarşisi

CObject

CWnd

İlk nesil

CDragWnd, CWndContainer, CWndObj

Sınıf Yöntemleri

Oluşturma ve kaldırma	
Create	Kontrolü oluşturur
Destroy	Kontrolü yok eder
Çizelge olay işleyicileri	
OnEvent	Tüm çizelge olayları için olay işleyicisi
OnMouseEvent	CHARTEVENT_MOUSE_MOVE (fare hareketi) olayı için olay işleyici
İsim	
Name	Kontrol ismini alır
Taşıyıcıya erişim	
ControlsTotal	Taşıyıcı içindeki kontrollerin sayısını alır
Control	Kontrolü indis numarasına göre alır
ControlFind	Kontrolü tanımlayıcı numarasına göre alır
Geometri	
Rect	CRect sınıf nesnesinin işaretçisini alır
Left	Sol-üst köşenin X koordinatını alır/ayarlar

Oluşturma ve kaldırma	
Top	Sol-üst köşenin Y koordinatını alır/ayarlar
Right	Sağ-alt köşenin X koordinatını alır/ayarlar
Bottom	Sağ-alt köşenin Y koordinatını alır/ayarlar
Width	Genişlik değerini alır/ayarlar
Height	Yükseklik değerini alır/ayarlar
Move	Kontrol için yeni koordinatlar ayarlar.
Shift	Kontrol koordinatlarını göreli olarak hareket ettirir
Resize	Kontrolün genişliğini/yüksekliğini ayarlar
Contains	Kontrolün/noktanın kontrol alanı içinde olduğunu doğrular
Hizalama	
Alignment	Kontrolün hizalama özelliklerini ayarlar
Align	Kontrol hizalamasını gerçekleştirir
Tanımlama	
Id	Kontrol tanımlayıcısını alır/ayarlar
Durum	
IsEnabled	Kontrolün etkinleştirilip etkinleştirilmediğini sorgular
Enable	Kontrolün etkinleştirilme durumunun değerini ayarlar
Disable	Kontrolü devre-dışı bırakır
IsVisible	Görünürlük bayrağını denetler
Visible	Görünürlük bayrağını ayarlar
Show	Kontrolü gösterir
Hide	Kontrolü gizler
IsActive	Kontrolün etkinliğini denetler
Activate	Kontrolü aktif hale getirir
Deactivate	Kontrolü pasif hale getirir
Durum bayrakları	
StateFlags	Kontrolün durum bayraklarını alır/ayarlar
StateFlagsSet	Kontrolün durum bayraklarını ayarlar
StateFlagsReset	Kontrolün durum bayraklarını sıfırlar
Özellik bayrakları	

Oluşturma ve kaldırma	
PropFlags	Kontrolün özellik bayraklarını alır/ayarlar
PropFlagsSet	Kontrolün özellik bayraklarını ayarlar
PropFlagsReset	Kontrolün özellik bayraklarını sıfırlar
Fare işlemleri	
MouseX	Fare konumunun X koordinatını alır/kaydeder
MouseY	Fare konumunun Y koordinatını alır/kaydeder
MouseFlags	Fare tuşlarının durumunu alır/kaydeder
MouseFocusKill	Fare odağını sonlandırır
İçsel olay işleyicileri	
OnCreate	"Create" (oluşturma) olayının işleyicisi
OnDestroy	"Destroy" (yok et) olayının işleyicisi
OnMove	"Move" (taşıma) olayının işleyicisi
OnResize	"Resize" (boyutlandırma) olayının işleyicisi
OnEnable	"Enable" (etkinleştir) olayının olay işleyicisi
OnDisable	"Disable" (devre-dışı bırak) olayının olay işleyicisi
OnShow	"Show" (gösterme) olayının işleyicisi
OnHide	"Hide" (gizleme) olayının işleyicisi
OnActivate	"Activate" (aktifleştir) olayının işleyicisi
OnDeactivate	"Deactivate" (pasifleştir) olayının işleyicisi
OnClick	"Click" (tıklama) olayının işleyicisi
OnChange	"Change" (değişim) olayının işleyicisi
Fare olaylarının işleyicileri	
OnMouseDown	"MouseDown" (fare tuşu basılı) olayının işleyicisi
OnMouseUp	"MouseUp" (fare tuşu serbest) olayının işleyicisi
Sürükleme olayı işleyicileri	
OnDragStart	"DragStart" (sürükleme başlangıcı) olayının işleyicisi
OnDragProcess	"DragProcess" (sürükleme) olayının işleyicisi
OnDragEnd	"DragEnd" (sürükleme sonu) olayının işleyicisi
Sürükleme nesneleri	
DragObjectCreate	Sürükleme nesnesi oluşturur

Oluşturma ve kaldırma	
DragObjectDestroy	Sürükleme nesnesini yok eder

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Create

Bir kontrol oluşturur.

```
virtual bool  Create(
   const long    chart,      // çizelge tanımlayıcısı
   const string  name,       // isim
   const int     subwin,     // çizelge alt-penceresi
   const int     x1,         // x1 koordinatı
   const int     y1,         // y1 koordinatı
   const int     x2,         // x2 koordinatı
   const int     y2          // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi sadece parametreleri kaydeder ve her zaman 'true' dönüşü yapar.

Destroy

Kontrolü yok eder.

```
virtual bool  Destroy()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnEvent

Çizelge olayı işleyicisi.

```
virtual bool  OnEvent(
   const int      id,          // tanımlayıcı
   const long&    lparam,      // long tipli olay parametresi
   const double&  dparam,      // double tipli olay parametresi
   const string&  sparam       // string tipli olay parametresi
   )
```

Parametreler

id

[in] Olay tanımlayıcısı.

lparam

[in] Referansla geçirilen, long tipli olay parametresi.

dparam

[in] Referansla geçirilen, double tipli olay parametresi.

sparam

[in] Referansla geçirilen, string tipli olay parametresi.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnMouseEvent

Fare olayı (CHARTEVENT_MOUSE_MOVE çizelge olayı) için sanal olay işleyici.

```
virtual bool  OnMouseEvent(
   const int  x,          // x koordinatı
   const int  y,          // y koordinatı
   const int  flags       // bayraklar
   )
```

Parametreler

x

[in] Fare işaretçisinin, çizelgenin sol-üst köşesine göre X koordinatı.

y

[in] Fare işaretçisinin, çizelgenin sol-üst köşesine göre Y koordinatı.

flags

[in] Fare tuşlarının durum bayrakları.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Name

Kontrol ismini alır.

```
string  Name()  const
```

Dönüş değeri

Kontrol ismi.

ControlsTotal

Taşıyıcıdaki kontrollerin sayısını alır.

```
int  ControlsTotal()  const
```

Dönüş değeri

Taşıyıcıdaki kontrollerin sayısı.

Not

Temel sınıf taşıyıcıya sahip değildir, soyundan gelen sınıflar için taşıyıcıya erişim sağlar ve daima 0 dönüşü yapar.

Control

Kontrol işaretçisini indisine göre alır.

```
CWnd*  Control(
   const int  ind      // indis
   )  const
```

Parametreler

ind

[in] Kontrol indisi.

Dönüş değeri

Kontrolün işaretçisi.

Not

Temel sınıf taşıyıcıya sahip değildir, soyundan gelen sınıflar için taşıyıcıya erişim sağlar ve daima NULL dönüşü yapar.

ControlFind

Belirtilen tanımlayıcıya göre kontrolü taşıyıcıdan alır.

```
virtual CWnd*  ControlFind(
   const long  id      // tanımlayıcı
   )
```

Parametreler

id

[in] Bulunacak kontrolün tanımlayıcısı.

Dönüş değeri

Kontrolün işaretçisi.

Not

Temel sınıf taşıyıcıya sahip değildir, soyundan gelen sınıflar için taşıyıcıya erişim sağlar. Belirtilen tanımlayıcı, taşıyıcının tanımlayıcısıyla uyuşuyorsa, kendi işaretçisine (this) dönüş yapar.

Rect

CRect çizelge nesnesinin işaretçisini alır.

```
const CRect*  Rect()  const
```

Dönüş değeri

CRect çizelge nesnesinin işaretçisi.

Left (Get Yöntemi)

Kontrolün sol üst köşesinin X koordinatını alır.

```
int  Left()
```

Dönüş değeri

Kontrolün sol üst köşesinin X koordinatı.

Left (Set Yöntemi)

Kontrolün sol üst köşesinin X koordinatını ayarlar.

```
void  Left(
   const int  x      // yeni x koordinatı
   )
```

Parametreler

x

[in] Sol-üst köşenin yeni X koordinatı.

Dönüş değeri

Yok.

Top (Get Yöntemi)

Kontrolün sol üst köşesinin Y koordinatını alır.

```
int  Top()
```

Dönüş değeri

Kontrolün sol üst köşesinin Y koordinatı.

Top (Set Yöntemi)

Kontrolün sol üst köşesinin Y koordinatını ayarlar.

```
void  Top(
   const int  y      // y koordinatı
   )
```

Parametreler

y

[in] Sol-üst köşenin Y koordinatı için yeni değer.

Dönüş değeri

Yok.

Right (Get Yöntemi)

Kontrolün sağ-alt köşesinin X koordinatını alır.

```
int  Right()
```

Dönüş değeri

Sağ-alt köşenin X koordinatı.

Right (Set Yöntemi)

Kontrolün sağ-alt köşesinin Y koordinatını ayarlar.

```
void  Right(
   const int  x      // x koordinatı
   )
```

Parametreler

x

[in] Sağ-alt köşe için yeni X koordinatı.

Dönüş değeri

Yok.

Bottom (Get Yöntemi)

Kontrolün sağ-alt köşesinin Y koordinatını alır.

```
int  Bottom()
```

Dönüş değeri

Kontrolün sağ-alt köşesinin Y koordinatını.

Bottom (Set Yöntemi)

Kontrolün sağ-alt köşesinin Y koordinatını ayarlar.

```
void  Bottom(
   const int  y      // y koordinatı
   )
```

Parametreler

y

[in] Sağ-alt köşenin Y koordinatı için yeni değer.

Dönüş değeri

Yok.

Width (Get Yöntemi)

Kontrolün genişlik değerini alır.

```
int  Width()
```

Dönüş değeri

Kontrolün genişlik değeri.

Width (Set Yöntemi)

Kontrolün genişlik değerini ayarlar.

```
virtual bool  Width(
   const int  w      // genişlik
   )
```

Parametreler

w

[in] Yeni genişlik değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Height (Get Yöntemi)

Kontrolün yükseklik değerini alır.

```
int  Height()
```

Dönüş değeri

Kontrolün yüksekliği.

Height (Set Yöntemi)

Kontrolün yükseklik değerini ayarlar.

```
virtual bool  Height(
   const int  h      // yükseklik
   )
```

Parametreler

h

[in] Yeni yükseklik.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Move

Kontrol için yeni koordinatlar ayarlar.

```
void  Move(
   const int  x,      // X koordinatı
   const int  y       // Y koordinatı
   )
```

Parametreler

x

[in] Yeni X koordinatı.

y

[in] Yeni Y koordinatı.

Dönüş değeri

Yok.

Shift

Kontrol koordinatlarını göreli olarak hareket ettirir.

```
void  Shift(
   const int  dx,      // delta X
   const int  dy       // delta Y
   )
```

Parametreler

dx

[in] Delta X.

dy

[in] Delta Y.

Dönüş değeri

Yok.

Resize

Kontrol için yeni yükseklik/genişlik ayarlar.

```
virtual bool  Resize(
   const int  w,      // genişlik
   const int  h       // yükseklik
   )
```

Parametreler

w

[in] Yeni genişlik değeri.

h

[in] Yeni yükseklik.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Contains

Noktanın kontrol alanı içinde olduğunu doğrular.

```
bool  Contains(
   const int  x,      // X koordinatı
   const int  y       // Y koordinatı
   )
```

Parametreler

x

[in] X koordinatı.

y

[in] Y koordinatı.

Dönüş değeri

Nokta, alanın içindeyse 'true', aksi durumda 'false'..

Contains

Belirtilen kontrolün, kontrol alanı içinde olduğunu doğrular.

```
bool  Contains(
   const CWnd*  control      // işaretçi
   )  const
```

Parametreler

control

[in] Nesne işaretçisi.

Dönüş değeri

Belirtilen kontrol, kontrol alanı (kenarlıklar dahil) içinde ise 'true', değil ise 'false'.

Alignment

Kontrolün hizalama parametrelerini ayarlar.

```
void  Alignment(
   const int  flags,       // hizalama bayrakları
   const int  left,        // sol girinti
   const int  top,         // üst girinti
   const int  right,       // sağ girinti
   const int  bottom       // alt girinti
   )
```

Parametreler

flags

[in] Hizalama bayrakları.

left

[in] Sol kenar sabit girintisi.

top

[in] Üst kenar sabit girintisi.

right

[in] Sağ kenar sabit girintisi.

bottom

[in] Alt kenar sabit girintisi.

Dönüş değeri

Yok.

Not

Hizalama bayrakları:

```
enum WND_ALIGN_FLAGS
{
   WND_ALIGN_NONE=0,                                  // hizalama yok
   WND_ALIGN_LEFT=1,                                  // sola hizala
   WND_ALIGN_TOP=2,                                   // üste hizala
   WND_ALIGN_RIGHT=4,                                 // sağa hizala
   WND_ALIGN_BOTTOM=8,                                // alta hizala
   WND_ALIGN_WIDTH = WND_ALIGN_LEFT|WND_ALIGN_RIGHT,  // hizalama genişliği
   WND_ALIGN_HEIGHT=WND_ALIGN_TOP|WND_ALIGN_BOTTOM,   // hizalama yüksekliği
   WND_ALIGN_CLIENT=WND_ALIGN_WIDTH|WND_ALIGN_HEIGHT, // hizalama yüksekliği ve genişl
   }
```

Align

Belirtilen çizelge alanında hizalama gerçekleştirir.

```
virtual bool  Align(
   const CRect*  rect      // işaretçi
   )
```

Parametreler

rect

[in] Çizelge alanı koordinatlarıyla nesne işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Hizalama parametreleri belirtilmelidir (varsayılan durumda hizalama yapılmaz).

Id (Get Yöntemi)

Kontrolün tanımlayıcısını alır.

```
long  Id()  const
```

Dönüş değeri

Kontrol tanımlayıcısı.

Id (Set Yöntemi)

Kontrol tanımlayıcısı için yeni değer ayarlar.

```
virtual long  Id(
   const long  id      // tanımlayıcı
   )
```

Parametreler

id

[in] Kontrol tanımlayıcısının yeni değeri.

Dönüş değeri

Yok.

IsEnabled

Kontrolün etkin durumda olup olmadığına dair bilgi alır.

```
bool  IsEnabled()  const
```

Dönüş değeri

Kontrol etkinleştirilmiş ise 'true', aksi durumda 'false'.

Enable

Kontrolü etkinleştirir.

```
virtual bool  Enable()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Etkinleştirilen kontroller dışsal olayları işleyebilirler.

Disable

Kontrolü devre-dışı bırakır.

```
virtual bool  Disable()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Devre-dışı bırakılan kontroller dışsal olayları işleyemezler.

IsVisible

Kontrolün görünür durumda olup olmadığına dair bilgi alır.

```
bool  IsVisible()  const
```

Dönüş değeri

Kontrol gösteriliyorsa 'true', aksi durumda 'false'.

Visible

Görünürlük bayrağını ayarlar.

```
virtual bool  Visible(
   const bool  flag      // bayrak
   )
```

Parametreler

flag

[in] Yeni bayrak.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Show

Kontrolü gösterir.

```
virtual bool  Show()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Hide

Kontrolü gizler.

```
virtual bool  Hide()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

IsActive

Kontrolün aktif olup olmadığına dair bilgi alır.

```
bool  IsActive()  const
```

Dönüş değeri

Kontrol aktifse 'true', aksi durumda 'false'.

Activate

Kontrolü aktif hale getirir.

```
virtual bool  Activate()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Kontrol, fare işaretçisi ile üzerinde gezinildiğinde aktif hale gelir.

Deactivate

Kontrolü pasif hale getirir.

```
virtual bool  Deactivate()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Fare işaretçisi kontrolün üzerinden ayrıldığında kontrol pasif durum geçer.

StateFlags (Get Yöntemi)

Kontrol durum bayraklarını alır.

```
int  StateFlags()
```

Dönüş değeri

Kontrol durum bayrakları.

StateFlags (Set Yöntemi)

Kontrol durum bayraklarını ayarlar.

```
virtual void  StateFlags(
   const int  flags      // bayraklar
   )
```

Parametreler

flags

[in] Yeni kontrol durum bayrakları.

Dönüş değeri

Yok.

StateFlagsSet

Kontrol durum bayraklarını ayarlar.

```
virtual void  StateFlagsSet(
   const int  flags      // bayraklar
   )
```

Parametreler

flags

[in] Ayarlanacak bayraklar (bit maskesi).

Dönüş değeri

Yok.

StateFlagsReset

Kontrol durum bayraklarını sıfırlar.

```
virtual void  StateFlagsReset(
   const int  flags      // bayraklar
   )
```

Parametreler

flags

[in] Sıfırlanacak bayraklar (bit maskesi).

Dönüş değeri

Yok.

PropFlags (Get Yöntemi)

Kontrol özelliklerinin bayraklarını alır.

```
void  PropFlags(
   const int  flags      // bayraklar
   )
```

Dönüş değeri

Kontrol özelliklerinin bayrakları.

PropFlags (Set Yöntemi)

Kontrol özelliklerinin bayraklarını ayarlar.

```
virtual void  PropFlags(
   const int  flags      // bayraklar
   )
```

Parametreler

flags

[in] Yeni bayraklar.

Dönüş değeri

Yok.

PropFlagsSet

Kontrol özelliklerinin bayraklarını ayarlar.

```
virtual void  PropFlagsSet(
   const int  flags      // bayraklar
   )
```

Parametreler

flags

[in] Ayarlanacak bayraklar (bit maskesi).

Dönüş değeri

Yok.

PropFlagsReset

Kontrol özelliklerinin bayraklarını sıfırlar.

```
virtual void  PropFlagsReset(
   const int  flags      // bayraklar
   )
```

Parametreler

flags

[in] Sıfırlanacak bayraklar (bit maskesi).

Dönüş değeri

Yok.

MouseX (Set Yöntemi)

Fare konumunun X koordinatını kaydeder

```
void  MouseX(
   const int  value      // koordinat
   )
```

Parametreler

value

[in] Fare işaretçisinin X koordinatı.

Dönüş değeri

Yok.

MouseX (Get Yöntemi)

Fare işaretçisinin X koordinatını alır.

```
int  MouseX()
```

Dönüş değeri

Fare işaretçisinin kaydedilen X koordinatı.

MouseY (Set Yöntemi)

Fare konumunun Y koordinatını kaydeder

```
void  MouseY(
   const int  value      // koordinat
   )
```

Parametreler

value

[in] Fare konumunun Y koordinatı.

Dönüş değeri

Yok.

MouseY (Get Yöntemi)

Fare konumunun kaydedilen Y koordinatını alır.

```
int  MouseY()
```

Dönüş değeri

Fare konumunun kaydedilen Y koordinatı.

MouseFlags (Set Yöntemi)

Fare tuşlarının durumunu kaydeder.

```
virtual void  MouseFlags(
   const int  value      // durum
   )
```

Parametreler

value

[in] Fare tuşlarının durumu.

Dönüş değeri

Yok.

MouseFlags (Get Yöntemi)

Fare tuşlarının kaydedilen durumunu alır.

```
int  MouseFlags()
```

Dönüş değeri

Fare tuşlarının durumu.

MouseFocusKill

Fare tuşlarının kaydedilen durumunu siler ve kontrolü pasif hale getirir.

```
bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // tanımlayıcı
   )
```

Parametreler

id=CONTROLS_INVALID_ID

[in] Fare odağındaki kontrolün tanımlayıcısı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnCreate

Kontrolün "Create" (oluşum) olayı için sanal olay işleyicisi.

```
virtual bool  OnCreate()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnDestroy

Kontrolün "Destroy" (yok etme) olayının sanal olay işleyicisi.

```
virtual bool  OnDestroy()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnMove

Kontrolün "Move" (taşıma) olayı için sanal olay işleyici.

```
virtual bool  OnMove()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnResize

Kontrolün "Resize" (boyutlandırma) olayı için sanal olay işleyici.

```
virtual bool  OnResize()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnEnable

Kontrolün "Enable" (etkinleştirilme) olayı için sanal olay işleyici.

```
virtual bool  OnEnable()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnDisable

Kontrolün "Disable" (devre-dışı bırakma) olayı için sanal olay işleyici.

```
virtual bool  OnDisable()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnShow

Kotrolün "Show" (göster) olayı için sanal olay işleyici.

```
virtual bool  OnShow()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnHide

Kontrolün "Hide" (gizle) olayı için sanal olay işleyici.

```
virtual bool  OnHide()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnActivate

Kontrolün "Activate" (aktifleşme) olayı için sanal olay işleyicisi.

```
virtual bool  OnActivate()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnDeactivate

Kontrolün "Deactivate" (pasif duruma geçme) olayı için sanal olay işleyicisi

```
virtual bool  OnDeactivate()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnClick

Kontrolün "Click" (sol fare tuşu tıklaması) olayı için sanal olay işleyici.

```
virtual bool  OnClick()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnChange

Kontrolün "Change" (değişim) olayı için sanal olay işleyicisi.

```
virtual bool  OnChange()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnMouseDown

Kontrolün "MouseDown" (fare tuşu basılı) olayı için sanal olay işleyici.

```
virtual bool  OnMouseDown()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"MouseDown" olayı, sol fare tuşu kontrol üzerinde basılı iken gerçekleşir.

OnMouseUp

Kontrolün "MouseUp" (fare tuşu serbest) olayı için sanal olay işleyici.

```
virtual bool  OnMouseUp()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"MouseUp" olayı, sol fare tuşu kontrol üzerinde serbest bırakıldığında gerçekleşir.

OnDragStart

Kontrolün "DragStart" (sürükleme başlangıcı) olayı için sanal olay işleyici.

```
virtual bool  OnDragStart()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"DragStart" olayı sürükleme işleminin başlangıcında ortaya çıkar.

OnDragProcess

Kontrolün "DragProcess" (sürükleme) olayı için sanal olay işleyici.

```
virtual bool  OnDragProcess(
   const int  x,      // x koordinatı
   const int  y       // y koordinatı
   )
```

Parametreler

x

[in] Fare işaretçisinin mevcut X koordinatı.

y

[in] Fare işaretçisinin mevcut Y koordinatı.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"DragProcess" olayı kontrol taşındığında oluşur.

OnDragEnd

Kontrolün "DragEnd" (sürükleme sonu) olayı için sanal olay işleyici.

```
virtual bool  OnDragEnd()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"DragEnd" olayı sürükleme işlemi bittiğinde oluşur.

DragObjectCreate

Sürükleme nesnesi oluşturur.

```
virtual bool  DragObjectCreate()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Olay işlenmişse 'true', aksi durumda 'false'.

DragObjectDestroy

sürükleme nesnesini yok eder.

```
virtual bool  DragObjectDestroy()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CWndObj

CWndObj sınıfı, Standart Kütüphanedeki — çizelge nesnelerine dayanan — basit kontroller için bir temel sınıftır.

Açıklama

CWndObj sınıfı, basit kontrollerin yöntemlerini uygular.

Bildirim

```
class CWndObj : public CWnd
```

Başlık

```
#include <Controls\WndObj.mqh>
```

Kalıtım hiyerarşisi

CObject
  CWnd
    CWndObj

İlk nesil

CBmpButton, CButton, CEdit, CLabel, CPanel, CPicture

Sınıf Yöntemleri

Çizelge olaylarının işlenmesi	
OnEvent	Tüm çizelge olayları için olay işleyicisi
Özellikler	
Metin	Çizelge nesnesinin OBJPROP_TEXT (nesne metni) özelliğinin değerini alır/ayarlar
Color	Çizelge nesnesinin OBJPROP_COLOR (nesne rengi) özelliğinin değerini alır/ayarlar
ColorBackground	Çizelge nesnesinin OBJPROP_BGCOLOR (nesne arka-plan rengi) özelliğinin değerini alır/ayarlar
ColorBorder	Çizelge nesnesinin OBJPROP_BORDER_COLOR (nesne kenarlık rengi) özelliğinin değerini alır/ayarlar
Font	Çizelge nesnesinin OBJPROP_FONT (nesne yazı-tipi) özelliğinin değerini alır/ayarlar
FontSize	Çizelge nesnesinin OBJPROP_FONTSIZE (nesne yazı-tipi boyutu) özelliğinin değerini alır/ayarlar
ZOrder	Çizelge nesnesinin OBJPROP_ZORDER (nesne tıklama önceliği) özelliğinin değerini alır/ayarlar

Çizelge olaylarının işlenmesi	
Çizelge nesnelerinin olay işleyicileri	
OnObjectCreate	CHARTEVENT_OBJECT_CREATE (oluşturma) olay işleyicisi
OnObjectChange	CHARTEVENT_OBJECT_CHANGE (değişim) olay işleyicisi
OnObjectDelete	CHARTEVENT_OBJECT_DELETE (silinme) olay işleyicisi
OnObjectDrag	CHARTEVENT_OBJECT_DRAG (sürükleme) olay işleyicisi
Özellik değişimlerinin olay işleyicileri	
OnSetText	"SetText" (Metin değişimi) olayı işleyicisi
OnSetColor	"SetColor" (renk değişimi) olayı işleyicisi
OnSetColorBackground	"SetColorBackground" (arka-plan değişimi) olayı işleyicisi
OnSetFont	"SetFont" (yazı-tipi değişimi) olayı işleyicisi
OnSetFontSize	"SetFontSize" (yazı-tipi boyutu değişimi) olayı işleyicisi
OnSetZOrder	"SetZOrder" (tıklama önceliği) olay işleyicisi
İçsel olay işleyicileri	
OnDestroy	"Destroy" (yok et) olayının işleyicisi
OnChange	"Change" (değişim) olayının işleyicisi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler 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

Çizelge olayı işleyicisi.

```
virtual bool  OnEvent(
   const int      id,          // tanımlayıcı
   const long&    lparam,      // long tipli olay parametresi
   const double&  dparam,      // double tipli olay parametresi
   const string&  sparam       // string tipli olay parametresi
   )
```

Parametreler

id

[in] Olay tanımlayıcısı.

lparam

[in] Referansla geçirilen long tipli olay parametresi.

dparam

[in] Referansla geçirilen double tipli olay parametresi.

sparam

[in] Referansla geçirilen string tipli olay parametresi.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Text (Get Yöntemi)

Çizelge nesnesinin OBJPROP_TEXT (metin) özelliğinin değerini alır.

```
string  Text()
```

Dönüş değeri

OBJPROP_TEXT özelliğinin değeri.

Text (Set Yöntemi)

Çizelge nesnesinin OBJPROP_TEXT (metin) özelliğinin değerini ayarlar.

```
bool  Text(
   const string  value      // yeni değer
   )
```

Parametreler

value

[in] OBJPROP_TEXT özelliğinin yeni değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Color (Get Yöntemi)

Çizelge nesnesinin OBJPROP_COLOR (renk) özelliğinin değerini alır.

```
color  Color()
```

Dönüş değeri

OBJPROP_COLOR özelliğinin değeri.

Color (Set Yöntemi)

Çizelge nesnesinin OBJPROP_COLOR (renk) özelliğinin değerini ayarlar.

```
bool  Color(
   const color  value      // değer
   )
```

Parametreler

value

[in] OBJPROP_COLOR özelliğinin değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

ColorBackground (Get Yöntemi)

Çizelge nesnesinin OBJPROP_BGCOLOR (arka-plan rengi) özelliğinin değerini alır.

```
color  ColorBackground()
```

Dönüş değeri

OBJPROP_BGCOLOR özelliğinin değeri.

ColorBackground (Set Yöntemi)

Çizelge nesnesinin OBJPROP_BGCOLOR (arka-plan rengi) özelliğinin değerini ayarlar.

```
bool  ColorBackground(
   const color  value      // değer
   )
```

Parametreler

value

[in] OBJPROP_BGCOLOR özelliğinin yeni değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

ColorBorder (Get Yöntemi)

Çizelge nesnesinin OBJPROP_BORDER_COLOR (kenarlık rengi) özelliğinin değerini alır.

```
color  ColorBorder()
```

Dönüş değeri

OBJPROP_BORDER_COLOR özelliğinin değeri.

ColorBorder (Set Yöntemi)

Çizelge nesnesinin OBJPROP_BORDER_COLOR (kenarlık rengi) özelliğinin değerini ayarlar.

```
bool  ColorBorder(
   const color  value      // değer
   )
```

Parametreler

value

[in] OBJPROP_BORDER_COLOR özelliğinin yeni değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Font (Get Yöntemi)

Çizelge nesnesinin OBJPROP_FONT (yazı-tipi) özelliğinin değerini alır.

```
string  Font()
```

Dönüş değeri

OBJPROP_FONT özelliğinin değeri.

Font (Set Yöntemi)

Çizelge nesnesinin OBJPROP_FONT (yazı-tipi) özelliğinin değerini ayarlar.

```
bool  Font(
   const string  value      // yeni değer
   )
```

Parametreler

value

[in] OBJPROP_FONT özelliğinin yeni değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

FontSize (Get Yöntemi)

Çizelge nesnesinin OBJPROP_FONTSIZE (yazı-tipi boyutu) özelliğinin değerini alır.

```
int  FontSize()
```

Dönüş değeri

OBJPROP_FONTSIZE özelliğinin değeri.

FontSize (Set Yöntemi)

Çizelge nesnesinin OBJPROP_FONTSIZE (yazı-tipi boyutu) özelliğinin değerini ayarlar.

```
bool  FontSize(
   const int  value      // yeni değer
   )
```

Parametreler

value

[in] OBJPROP_FONTSIZE özelliğinin yeni değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

ZOrder (Get Yöntemi)

Çizelge nesnesinin OBJPROP_ZORDER (tıklama önceliği) özelliğinin değerini alır.

```
long  ZOrder()
```

Dönüş değeri

OBJPROP_ZORDER property.

ZOrder (Set Yöntemi)

Çizelge nesnesinin OBJPROP_ZORDER (tıklama önceliği) özelliğinin değerini ayarlar.

```
bool  ZOrder(
   const long  value      // yeni değer
   )
```

Parametreler

value

[in] OBJPROP_ZORDER özelliğinin yeni değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnObjectCreate

Çizelge nesnesinin CHARTEVENT_OBJECT_CREATE (oluşturma) olayı işleyicisi.

```
virtual bool  OnObjectCreate()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnObjectChange

Çizelge nesnesinin CHARTEVENT_OBJECT_CHANGE (değişim) olayı işleyicisi.

```
virtual bool  OnObjectChange()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnObjectDelete

Çizelge nesnesinin CHARTEVENT_OBJECT_DELETE (silme) olayı işleyicisi.

```
virtual bool  OnObjectDelete()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnObjectDrag

Çizelge nesnesinin CHARTEVENT_OBJECT_DRAG (sürükleme) olay işleyicisi.

```
virtual bool  OnObjectDrag()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetText

Kontrolün "SetText" (OBJPROP_TEXT özeliğinin değişimi) olayının işleyicisi.

```
virtual bool  OnSetText()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnSetColor

Kontrolün "SetColor" (OBJPROP_COLOR özelliğinin değişimi) olayı için sanal olay işleyicisi.

```
virtual bool  OnSetColor()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnSetColorBackground

Kontrolün "SetColorBackground" (OBJPROP_BGCOLOR özelliğinin değişimi) olayının işleyicisi.

```
virtual bool  OnSetColorBackground()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnSetFont

Kontrolün "SetFont" (OBJPROP_FONT özelliğinin değişimi) olayının işleyicisi.

```
virtual bool  OnSetFont()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnSetFontSize

Kontrolün "SetFontSize" (OBJPROP_FONTSIZE özelliğinin değişimi) olayının işleyicisi.

```
virtual bool  OnSetFontSize()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnSetZOrder

Kontrolün "SetZOrder" (OBJPROP_ZORDER özelliğinin değişimi) olayının işleyicisi.

```
virtual bool  OnSetZOrder()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnDestroy

Kontrolün "Destroy" (yok etme) olayının sanal olay işleyicisi.

```
virtual bool  OnDestroy()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnChange

Kontrolün "Change" (değişim) olayı için sanal olay işleyicisi.

```
virtual bool  OnChange()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

CWndContainer

CWndContainer, Standart Kütüphanedeki karmaşık kontroller için bir temel sınıfıdır.

Açıklama

CWndContainer sınıfı karmaşık kontrolün temel yöntemlerini uygular.

Bildirim

```
class CWndContainer : public CWnd
```

Başlık

```
#include <Controls\WndContainer.mqh>
```

Kalıtım hiyerarşisi

CObject

CWnd

CWndContainer

İlk nesil

CCheckBox, CComboBox, CDateDropList, CDatePicker, CDialog, CRadioButton, CScroll, CSpinEdit, CWndClient

Sınıf Yöntemleri

Yok et	
Destroy	Tüm taşıyıcı kontrollerini yok eder
Çizelge olay işleyicileri	
OnEvent	Tüm çizelge olayları için olay işleyicisi
OnMouseEvent	CHARTEVENT_MOUSE_MOVE olayının işleyicisi
Taşıyıcıya erişim	
ControlsTotal	Taşıyıcıda yer alan kontrollerin sayısını alır
Control	Kontrolü indis numarasına göre alır
ControlFind	Kontrolü tanımlayıcı numarasına göre alır
Ekle/Sil	
Add	Taşıyıcıya kontrol ekler
Delete	Taşıyıcıdaki kontrolleri yok eder
Geometri	
Move	Taşıyıcıdaki tüm kontroller için yeni koordinatlar ayarlar

Yok et	
Shift	Taşıyıcıdaki tüm kontrollerin koordinatlarını göreli olarak hareket ettirir
Tanımlama	
Id	Taşıyıcıdaki tüm kontroller için yeni tanımlayıcı ayarlar
Durum	
Enable	Taşıyıcıdaki tüm kontrolleri etkinleştirir
Disable	Taşıyıcıdaki tüm kontrolleri devre-dışı bırakır
Show	Taşıyıcıdaki tüm kontrolleri gösterir
Hide	Taşıyıcıdaki tüm kontrolleri gizler
Fare işlemleri	
MouseFocusKill	Fare odağını sonlandırır
Dosya işlemleri	
Save	Taşıyıcı bilgisini dosyaya kaydeder
Load	Taşıyıcı bilgisini dosyadan yükler
İçsel olay işleyicileri	
OnResize	"Resize" (boyutlandırma) olayının işleyicisi
OnActivate	"Activate" (aktifleştir) olayının işleyicisi
OnDeactivate	"Deactivate" (pasifleştir) olayının işleyicisi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

Sınıftan türetilen yöntemler 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

Tüm taşıyıcı kontrollerini yok eder.

```
virtual bool  Destroy()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnEvent

Çizelge olayı işleyicisi.

```
virtual bool  OnEvent(
   const int      id,          // tanımlayıcı
   const long&    lparam,      // long tipli olay parametresi
   const double&  dparam,      // double tipli olay parametresi
   const string&  sparam       // string tipli olay parametresi
   )
```

Parametreler

id

[in] Olay tanımlayıcısı.

lparam

[in] Referansla geçirilen long tipli olay parametresi.

dparam

[in] Referansla geçirilen double tipli olay parametresi.

sparam

[in] Referansla geçirilen string tipli olay parametresi.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnMouseEvent

Fare olayının işleyicisi.

```
virtual bool  OnMouseEvent(
   const int  x,          // x koordinatı
   const int  y,          // y koordinatı
   const int  flags       // bayraklar
   )
```

Parametreler

x

[in] Fare işaretçisinin, çizelgenin sol-üst köşesine göre X koordinatı.

y

[in] Fare işaretçisinin, çizelgenin sol-üst köşesine göre Y koordinatı.

flags

[in] Fare tuşlarının durum bayrakları.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

ControlsTotal

Taşıyıcıdaki kontrollerin sayısını alır.

```
int  ControlsTotal()  const
```

Dönüş değeri

Taşıyıcıdaki kontrollerin sayısı.

Control

Kontrolü indis numarasına göre taşıyıcıdan alır.

```
CWnd*  Control(
   const int  ind      // indis
   )  const
```

Parametreler

ind

[in] İstenen kontrolün indisi.

Dönüş değeri

Başarılı ise istenen kontrolün işaretçisine, kontrol bulunamadıysa NULL değerine dönüş yapar.

ControlFind

Belirtilen tanımlayıcıya göre kontrolü taşıyıcıdan alır.

```
virtual CWnd*  ControlFind(
   const long  id      // tanımlayıcı
   )
```

Parametreler

id

[in] İstenen kontrolün tanımlayıcısı.

Dönüş değeri

Başarılı ise istenen kontrolün işaretçisine, kontrol bulunamadıysa NULL değerine dönüş yapar.

Add

Taşıyıcıya bir kontrol ekler.

```
bool  Add(
   CWnd&  control      // referans
   )
```

Parametreler

control

[in] Referansla geçirilen, eklenecek kontrol.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Delete

Kontrolü taşıyıcıdan siler.

```
bool  Delete(
   CWnd&  control      // referans
   )
```

Parametreler

control

[in] Referansla geçirilen, silinecek kontrol.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Move

Taşıyıcıdaki tüm kontroller için yeni koordinatlar ayarlar.

```
virtual bool  Move(
   const int  x,      // x koordinatı
   const int  y       // y koordinatı
   )
```

Parametreler

x

[in] Sol-üst köşenin yeni X koordinatı.

y

[in] Sol-üst köşenin yeni Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Shift

Taşıyıcıdaki tüm kontrollerin koordinatlarını göreli olarak hareket ettirir.

```
virtual bool  Shift(
   const int  dx,      // x değişimi
   const int  dy       // y değişimi
   )
```

Parametreler

dx

[in] Delta X.

dy

[in] Delta Y.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Id

Taşıyıcıdaki tüm kontroller için yeni tanımlayıcı ayarlar.

```
virtual long  Id(
   const long  id      // tanımlayıcı
   )
```

Parametreler

id

[in] Temel grup tanımlayıcısı.

Dönüş değeri

Taşıyıcı kontrolleri tarafından kullanılan tanımlayıcıların sayısı.

Enable

Taşıyıcıdaki tüm kontrolleri etkinleştirir.

```
virtual bool  Enable()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Disable

Taşıyıcıdaki tüm kontrolleri devre-dışı bırakır.

```
virtual bool  Disable()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Show

Taşıyıcıdaki tüm kontrolleri gösterir.

```
virtual bool  Show()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Hide

Taşıyıcıdaki tüm kontrolleri gizler.

```
virtual bool  Hide()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

MouseFocusKill

Fare tuşlarının kaydedilen durumunu siler ve kontrolü pasif hale getirir.

```
bool  MouseFocusKill(
   const long  id=CONTROLS_INVALID_ID      // tanımlayıcı
   )
```

Parametreler

id=CONTROLS_INVALID_ID

[in] Fare odağındaki kontrolün tanımlayıcısı.

Dönüş değeri

Kontroller pasif hale getirilmişse 'true', aksi durumda 'false'.

Save

Taşıyıcı bilgisini dosyaya kaydeder.

```
virtual bool  Save(
   const int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] Yazma için açılacak ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Load

Taşıyıcı bilgisini dosyadan yükler.

```
virtual bool  Load(
   const int  file_handle      // dosya tanıtıcı değeri
   )
```

Parametreler

file_handle

[in] Okuma için açılacak ikili dosyanın tanıtıcı değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnResize

Kontrolün "Resize" (boyutlandırma) olayı için sanal olay işleyici.

```
virtual bool  OnResize()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnActivate

Kontrolün "Activate" (aktifleşme) olayı için sanal olay işleyicisi.

```
virtual bool  OnActivate()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnDeactivate

Kontrolün "Deactivate" (pasif duruma geçme) olayı için sanal olay işleyicisi

```
virtual bool  OnDeactivate()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

CLabel

CLabel "Metin etiketi" grafik nesnesine dayanan basit kontrol sınıfıdır.

Açıklama

CLabel, basit metin etiketleri oluşturmak için tasarlanmıştır.

Bildirim

```
class CLabel : public CWndObj
```

Başlık

```
#include <Controls\Label.mqh>
```

Kalıtım hiyerarşisi

CObject
  CWnd
    CWndObj
      CLabel

Kodun sonucu aşağıda verilmiştir:

Sınıf Yöntemleri

Oluştur	
Create	Kontrolü oluşturur

Oluştur	
Özellik değişimlerinin olay işleyicileri	
OnSetText	"SetText" (Metin değişimi) olayı işleyicisi
OnSetColor	"SetColor" (renk değişimi) olayı işleyicisi
OnSetFont	"SetFont" (yazı-tipi değişimi) olayı işleyicisi
OnSetFontSize	"SetFontSize" (yazı-tipi boyutu değişimi) olayı işleyicisi
İçsel olay işleyicileri	
OnCreate	"Create" (oluşturma) olayının işleyicisi
OnShow	"Show" (gösterme) olayının işleyicisi
OnHide	"Hide" (gizleme) olayının işleyicisi
OnMove	"Move" (taşıma) olayının işleyicisi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Metin etiketi ile panel oluşturma örneği:

```
//+------------------------------------------------------------------+
//|                                                ControlsLabel.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Kontrol Panelleri ve İletişim Kutuları. Gösterim sınıfı CLabel"
#include <Controls\Dialog.mqh>
#include <Controls\Label.mqh>
//+------------------------------------------------------------------+
//| tanımlar                                                         |
```

```
//+------------------------------------------------------------------+
//--- girintiler ve boşluklar
#define INDENT_LEFT                         (11)      // sol girinti (izin verilen çe
#define INDENT_TOP                          (11)      // üst girinti (izin verilen çe
#define INDENT_RIGHT                        (11)      // sağ girinti (izin verilen çe
#define INDENT_BOTTOM                       (11)      // alt girinti (izin verilen çe
#define CONTROLS_GAP_X                      (5)       // X koordinatıyla boşluk
#define CONTROLS_GAP_Y                      (5)       // Y kordinatıyla boşluk
//--- düğmeler için
#define BUTTON_WIDTH                        (100)     // X koordinatıyla genişlik
#define BUTTON_HEIGHT                       (20)      // Y koordinatıyla genişlik
//--- gösterge alanı için
#define EDIT_HEIGHT                         (20)      // Y koordinatıyla genişlik
//--- grup kontrolleri için
#define GROUP_WIDTH                         (150)     // X koordinatıyla genişlik
#define LIST_HEIGHT                         (179)     // Y koordinatıyla genişlik
#define RADIO_HEIGHT                        (56)      // Y koordinatıyla genişlik
#define CHECK_HEIGHT                        (93)      // Y koordinatıyla genişlik
//+------------------------------------------------------------------+
//| CControlsDialog Sınıfı                                           |
//| Kullanım: Kontroller uygulamasının ana iletişim kutusu           |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CLabel            m_label;                       // CLabel nesnesi
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- oluştur
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- çizelge olay işleyicisi
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
protected:
   //--- bağımlı kontroller oluştur
   bool              CreateLabel(void);
   //--- bağımlı olayların işleyicileri
   void              OnClickLabel(void);
  };
//+------------------------------------------------------------------+
//| Olay İşleme                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)

EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Yapıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
```

```
  {
  }
//+------------------------------------------------------------------+
//| Yıkıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Oluştur                                                          |
//+------------------------------------------------------------------+
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);
//--- bağımlı kontroller oluştur
   if(!CreateLabel())
      return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| "CLabel" oluştur                                                 |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateLabel(void)
  {
//--- koordinatlar
   int x1=INDENT_RIGHT;
   int y1=INDENT_TOP+CONTROLS_GAP_Y;
   int x2=x1+100;
   int y2=y1+20;
//--- oluştur
   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);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
```

```
  {
//--- Uygulama iletişim kutusunu göster
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- uygulamayı çalıştır
   ExtDialog.Run();
//--- başarılı
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- iletişim kutusunu yok et
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // olay tanıtıcısı
                  const long& lparam,   // long tipli olay parametresi
                  const double& dparam, // double tipli olay parametresi
                  const string& sparam) // string tipli olay parametresi
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Yeni bir CLabel kontrolü oluşturur.

```
virtual bool  Create(
   const long     chart,       // çizelge tanımlayıcısı
   const string   name,        // isim
   const int      subwin,      // çizelge alt-penceresi
   const int      x1,          // x1 koordinatı
   const int      y1,          // y1 koordinatı
   const int      x2,          // x2 koordinatı
   const int      y2           // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnSetText

Kontrolün "SetText" (OBJPROP_TEXT özelliğinin değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnSetText()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetColor

Kontrolün "SetColor" (OBJPROP_COLOR özelliğinin değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnSetColor()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetFont

Kontrolün "SetFont" (OBJPROP_FONT özelliğinin değişimi) olayı için sanal olay işleyici

```
virtual bool  OnSetFont()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetFontSize

Kontrolün "SetFontSize" (OBJPROP_FONTSIZE özelliğinin değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnSetFontSize()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnCreate

Kontrolün "Create" (oluşum) olayı için sanal olay işleyicisi.

```
virtual bool  OnCreate()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnShow

Kotrolün "Show" (göster) olayı için sanal olay işleyici.

```
virtual bool  OnShow()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnHide

Kontrolün "Hide" (gizle) olayı için sanal olay işleyici.

```
virtual bool  OnHide()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnMove

Kontrolün "Move" (taşıma) olayı için sanal olay işleyici.

```
virtual bool  OnMove()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

CBmpButton

CBmpButton, "Biteşlem etiketi" grafik nesnesi temelinde oluşturulmuş basit kontrol sınıfıdır.

Açıklama

CBmpButton, grafik resimler kullanarak düğmeler oluşturmak için tasarlanmıştır.

Bildirim

```
class CBmpButton : public CWndObj
```

Başlık

```
#include <Controls\BmpButton.mqh>
```

Kalıtım hiyerarşisi

CObject
  CWnd
    CWndObj
      CBmpButton

Kodun sonucu aşağıda verilmiştir:

Sınıf Yöntemleri

Oluştur	
Create	Kontrolü oluşturur

Oluştur	
Özellikler	
Border	Kontrolün "Kenarlık" özelliğini alır/ayarlar
BmpNames	Kontrolün bmp dosyasının ismini ayarlar
BmpOffName	Düğme serbest (OFF) durum için kullanılacak bmp dosyasının ismini alır/ayarlar
BmpOnName	Düğme basılı (ON) durum için kullanılacak bmp dosyasının ismini alır/ayarlar
BmpPassiveName	Pasif durum için kullanılacak bmp dosyasının ismini alır/ayarlar
BmpActiveName	Aktif durum için kullanılacak bmp dosyasının ismini alır/ayarlar
Durum	
Pressed	Kontrolün (basılı) olma durumu alır/ayarlar
Locking	Kontrolün "Locking" (kilit) özelliğini alır/ayarlar
İçsel olay işleyicileri	
OnSetZOrder	"SetZOrder" (tıklama önceliği) olay işleyicisi
OnCreate	"Create" (oluşturma) olayının işleyicisi
OnShow	"Show" (gösterme) olayının işleyicisi
OnHide	"Hide" (gizleme) olayının işleyicisi
OnMove	"Move" (taşıma) olayının işleyicisi
OnChange	"Change" (değişim) olayının işleyicisi
OnActivate	"Activate" (aktifleştir) olayının işleyicisi
OnDeactivate	"Deactivate" (pasifleştir) olayının işleyicisi
OnMouseDown	"MouseDown" (fare tuşu basılı) olayının işleyicisi
OnMouseUp	"MouseUp" (fare tuşu serbest) olayının işleyicisi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CWndObj

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Biteşlem etiketi ile panel oluşturma örneği:

```
//+------------------------------------------------------------------+
//|                                            ControlsBmpButton.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Kontrol Panelleri ve İletişim Kutuları. Gösterim sınıfı CBmpBut
#include <Controls\Dialog.mqh>
#include <Controls\BmpButton.mqh>
//+------------------------------------------------------------------+
//| tanımlar                                                         |
//+------------------------------------------------------------------+
//--- girintiler ve boşluklar
#define INDENT_LEFT                         (11)      // sol girinti (izin verilen çe
#define INDENT_TOP                          (11)      // üst girinti (izin verilen çe
#define INDENT_RIGHT                        (11)      // sağ girinti (izin verilen çe
#define INDENT_BOTTOM                       (11)      // alt girinti (izin verilen çe
#define CONTROLS_GAP_X                      (5)       // X koordinatıyla boşluk
#define CONTROLS_GAP_Y                      (5)       // Y kordinatıyla boşluk
//--- düğmeler için
#define BUTTON_WIDTH                        (100)     // X koordinatıyla genişlik
#define BUTTON_HEIGHT                       (20)      // Y koordinatıyla genişlik
//--- gösterge alanı için
#define EDIT_HEIGHT                         (20)      // Y koordinatıyla genişlik
//--- grup kontrolleri için
#define GROUP_WIDTH                         (150)     // X koordinatıyla genişlik
#define LIST_HEIGHT                         (179)     // Y koordinatıyla genişlik
#define RADIO_HEIGHT                        (56)      // Y koordinatıyla genişlik
#define CHECK_HEIGHT                        (93)      // Y koordinatıyla genişlik
//+------------------------------------------------------------------+
//| CControlsDialog Sınıfı                                           |
//| Kullanım: Kontroller uygulamasının ana iletişim kutusu           |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CBmpButton        m_bmpbutton1;                  // CBmpButton nesnesi
   CBmpButton        m_bmpbutton2;                  // CBmpButton nesnesi
```

```
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- oluştur
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- çizelge olay işleyicisi
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- bağımlı kontroller oluştur
   bool              CreateBmpButton1(void);
   bool              CreateBmpButton2(void);
   //--- bağımlı olayların işleyicileri
   void              OnClickBmpButton1(void);
   void              OnClickBmpButton2(void);
  };
//+------------------------------------------------------------------+
//| Olay İşleme                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_bmpbutton1,OnClickBmpButton1)
ON_EVENT(ON_CLICK,m_bmpbutton2,OnClickBmpButton2)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Yapıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Yıkıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Oluştur                                                          |
//+------------------------------------------------------------------+
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);
//--- bağımlı kontroller oluştur
   if(!CreateBmpButton1())
      return(false);
   if(!CreateBmpButton2())
      return(false);
//--- başarılı
   return(true);
```

```
  }
//+------------------------------------------------------------------+
//| "BmpButton1" düğmesini oluştur                                   |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateBmpButton1(void)
  {
//--- koordinatlar
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- oluştur
   if(!m_bmpbutton1.Create(m_chart_id,m_name+"BmpButton1",m_subwin,x1,y1,x2,y2))
      return(false);
//--- CBmpButton kkontrolünün bmp dosyalarının ismini ayarlar
   m_bmpbutton1.BmpNames("\\Images\\euro.bmp","\\Images\\dollar.bmp");
   if(!Add(m_bmpbutton1))
      return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| "BmpButton2" sabit düğmesini oluştur                             |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateBmpButton2(void)
  {
//--- koordinatlar
   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;
//--- oluştur
   if(!m_bmpbutton2.Create(m_chart_id,m_name+"BmpButton2",m_subwin,x1,y1,x2,y2))
      return(false);
//--- CBmpButton kkontrolünün bmp dosyalarının ismini ayarlar
   m_bmpbutton2.BmpNames("\\Images\\euro.bmp","\\Images\\dollar.bmp");
   if(!Add(m_bmpbutton2))
      return(false);
   m_bmpbutton2.Locking(true);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| Olay İşleyici                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickBmpButton1(void)
  {
   Comment(__FUNCTION__);
  }
```

```
//+------------------------------------------------------------------+
//| Olay İşleyici                                                    |
//+------------------------------------------------------------------+
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()
  {
//--- Uygulama iletişim kutusunu göster
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- uygulamayı çalıştır
   ExtDialog.Run();
//--- başarılı
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- iletişim kutusunu yok et
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // olay tanıtıcısı
                  const long& lparam,   // long tipli olay parametresi
                  const double& dparam, // double tipli olay parametresi
                  const string& sparam) // string tipli olay parametresi
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Yeni bir CBmpButton kontrolü oluşturur.

```
virtual bool  Create(
   const long     chart,      // çizelge tanımlayıcısı
   const string   name,       // isim
   const int      subwin,     // çizelge alt-penceresi
   const int      x1,         // x1 koordinatı
   const int      y1,         // y1 koordinatı
   const int      x2,         // x2 koordinatı
   const int      y2          // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Border (Get Yöntemi)

Kontrolün "Border" (kenarlık kalınlığı) özelliğini alır.

```
int  Border()  const
```

Dönüş değeri

"Border" özelliğinin değeri.

Border (Set Yöntemi)

Kontrolün "Border" (kenarlık kalınlığı) özelliğini ayarlar.

```
bool  Border(
   const int  value      // yeni değer
   )
```

Parametreler

value

[in] "Border özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

BmpNames

Kontrolün bmp dosyasının ismini ayarlar

```
bool  BmpNames(
   const string  off="",      // dosya ismi
   const string  on=""        // dosya ismi
   )
```

Parametreler

off=""

[in] Düğme serbest durum (OFF) için bmp dosyası ismi.

on=""

[in] Düğme basılı durum (ON) için bmp dosyası ismi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

BmpOffName (Get Yöntemi)

Düğme serbest durum (OFF) için kullanılacak bmp dosyasının ismini alır

```
string  BmpOffName()  const
```

Dönüş değeri

Düğme serbest durum (OFF) için bmp dosyası ismi.

BmpOffName (Set Yöntemi)

Düğme serbest durum (OFF) için kullanılacak bmp dosyasının ismini ayarlar

```
bool  BmpOffName(
   const string  name      // dosya ismi
   )
```

Parametreler

name

[in] Düğme serbest durum (OFF) için bmp dosyası ismi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

BmpOnName (Get Yöntemi)

Düğme basılı durum (ON) için kullanılacak bmp dosyasının ismini alır

```
string  BmpOnName()  const
```

Dönüş değeri

Düğme basılı durum (ON) için bmp dosyasının ismi

BmpOnName (Set Yöntemi)

Düğme basılı durum (ON) için kullanılacak bmp dosyasının ismini ayarlar.

```
bool  BmpOnName(
   const string  name      // dosya ismi
   )
```

Parametreler

name

[in] Düğme basılı durum (ON) için bmp dosyası ismi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

BmpPassiveName (Get Yöntemi)

Pasif durum için kullanılacak bmp dosyasının ismini alır.

```
string  BmpPassiveName()  const
```

Dönüş değeri

Pasif durum için bmp dosyası ismi.

BmpPassiveName (Set Yöntemi)

Pasif durum için kullanılacak bmp dosyasının ismini ayarlar.

```
bool  BmpPassiveName(
   const string  name      // dosya ismi
   )
```

Parametreler

name

[in] Pasif durum için bmp dosyası ismi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

BmpActiveName (Get Yöntemi)

Aktif durum için kullanılacak bmp dosyasının ismini alır.

```
string  BmpActiveName()  const
```

Dönüş değeri

Aktif durum için bmp dosyası ismi.

Not

Kontrol, fare işaretçisi ile üzerinde gezinildiğinde aktif hale gelir.

BmpActiveName (Set Yöntemi)

Aktif durum için kullanılacak bmp dosyasının ismini ayarlar.

```
bool  BmpActiveName(
   const string  name      // dosya ismi
   )
```

Parametreler

name

[in] Aktif durum için bmp dosyası ismi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Pressed (Get Yöntemi)

Kontrolün basılı olup olmama durumunu ("Pressed" özelliğini) alır.

```
bool  Pressed()  const
```

Dönüş değeri

Kontrol durumu.

Pressed (Set Yöntemi)

Kontrolün basılı olup olmama durumunu ("Pressed" özelliğini) ayarlar.

```
bool  Pressed(
   const bool  pressed      // yeni durum
   )
```

Parametreler

pressed

[in] Yeni kontrol durumu.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Locking (Get Yöntemi)

Kontrolün "Locking" (kilit) özelliğini alır.

```
bool  Locking()  const
```

Dönüş değeri

"Locking" özelliği için yeni değer.

Locking (Set Yöntemi)

Kontrolün "Locking" (kilit) özelliği için yeni değer ayarlar.

```
void  Locking(
   const bool  locking      // yeni değer
   )
```

Parametreler

locking

[in] "Locking" özelliğinin yeni değeri.

Dönüş değeri

Yok.

OnSetZOrder

Kontrolün "SetZOrder" (OBJPROP_ZORDER özelliğinin değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnSetZOrder()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnCreate

Kontrolün "Create" (oluşum) olayı için sanal olay işleyicisi.

```
virtual bool  OnCreate()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnShow

Kotrolün "Show" (göster) olayı için sanal olay işleyici.

```
virtual bool  OnShow()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnHide

Kontrolün "Hide" (gizle) olayı için sanal olay işleyici.

```
virtual bool  OnHide()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnMove

Kontrolün "Move" (taşıma) olayı için sanal olay işleyici.

```
virtual bool  OnMove()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnChange

Kontrolün "Change" (değişim) olayı için sanal olay işleyicisi.

```
virtual bool  OnChange()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnActivate

Kontrolün "Activate" (aktifleşme) olayı için sanal olay işleyicisi.

```
virtual bool  OnActivate()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnDeactivate

Kontrolün "Deactivate" (pasif duruma geçme) olayı için sanal olay işleyicisi

```
virtual bool  OnDeactivate()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnMouseDown

Kontrolün "MouseDown" (fare tuşu basılı) olayı için sanal olay işleyici.

```
virtual bool  OnMouseDown()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"MouseDown" olayı, sol fare tuşu kontrol üzerinde basılı iken gerçekleşir.

OnMouseUp

Kontrolün "MouseUp" (fare tuşu serbest) olayı için sanal olay işleyici.

```
virtual bool  OnMouseUp()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"MouseUp" olayı, sol fare tuşu kontrol üzerinde serbest bırakıldığında gerçekleşir.

CButton

CButton, "Düğme" grafik nesnesini temel alan bir basit kontrol sınıfıdır.

Açıklama

CButton sınıfı, basit düğmelerin oluşturulması için tasarlanmıştır.

Bildirim

```
class CButton : public CWndObj
```

Başlık

```
#include <Controls\Button.mqh>
```

Kalıtım hiyerarşisi

CObject
  CWnd
    CWndObj
      CButton

Kodun sonucu aşağıda verilmiştir:

Sınıf Yöntemleri

Oluşturma	
Create	Kontrolü oluşturur

Oluşturma	
Durum	
Pressed	"Pressed" (basılı) özelliğinin değerini alır/ayarlar
Locking	"Locking" (kilit) özelliğinin değerini alır/ayarlar
Özellik değişimlerinin olay işleyicileri	
OnSetText	"SetText" (Metin değişimi) olayı işleyicisi
OnSetColor	"SetColor" (renk değişimi) olayı işleyicisi
OnSetColorBackground	"SetColorBackground" (arka-plan değişimi) olayı işleyicisi
OnSetColorBorder	"SetColorBorder" (kenarlık rengi değişimi) olayı işleyicisi
OnSetFont	"SetFont" (yazı-tipi değişimi) olayı işleyicisi
OnSetFontSize	"SetFontSize" (yazı-tipi boyutu değişimi) olayı işleyicisi
İçsel olay işleyicileri	
OnCreate	"Create" (oluşturma) olayının işleyicisi
OnShow	"Show" (gösterme) olayının işleyicisi
OnHide	"Hide" (gizleme) olayının işleyicisi
OnMove	"Move" (taşıma) olayının işleyicisi
OnResize	"Resize" (boyutlandırma) olayının işleyicisi
OnMouseDown	"MouseDown" (fare tuşu basılı) olayının işleyicisi
OnOnMouseUp	"MouseUp" (fare tuşu serbest) olayının işleyicisi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Düğmeli panel oluşturma örneği

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

```
//|                                                ControlsButton.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Kontrol Panelleri ve İletişim Kutuları. Gösterim sınıfı CButtor
#include <Controls\Dialog.mqh>
#include <Controls\Button.mqh>
//+------------------------------------------------------------------+
//| tanımlar                                                         |
//+------------------------------------------------------------------+
//--- girintiler ve boşluklar
#define INDENT_LEFT                         (11)      // sol girinti (izin verilen çe
#define INDENT_TOP                          (11)      // üst girinti (izin verilen çe
#define INDENT_RIGHT                        (11)      // sağ girinti (izin verilen çe
#define INDENT_BOTTOM                       (11)      // alt girinti (izin verilen çe
#define CONTROLS_GAP_X                      (5)       // X koordinatıyla boşluk
#define CONTROLS_GAP_Y                      (5)       // Y kordinatıyla boşluk
//--- düğmeler için
#define BUTTON_WIDTH                        (100)     // X koordinatıyla genişlik
#define BUTTON_HEIGHT                       (20)      // Y koordinatıyla genişlik
//--- gösterge alanı için
#define EDIT_HEIGHT                         (20)      // Y koordinatıyla genişlik
//--- grup kontrolleri için
#define GROUP_WIDTH                         (150)     // X koordinatıyla genişlik
#define LIST_HEIGHT                         (179)     // Y koordinatıyla genişlik
#define RADIO_HEIGHT                        (56)      // Y koordinatıyla genişlik
#define CHECK_HEIGHT                        (93)      // Y koordinatıyla genişlik
//+------------------------------------------------------------------+
//| CControlsDialog Sınıfı                                           |
//| Kullanım: Kontroller uygulamasının ana iletişim kutusu           |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CButton           m_button1;                     // düğme nesnesi
   CButton           m_button2;                     // düğme nesnesi
   CButton           m_button3;                     // sabit düğme nesnesi

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- oluştur
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- çizelge olay işleyicisi
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons
```

```
protected:
   //--- bağımlı kontroller oluştur
   bool              CreateButton1(void);
   bool              CreateButton2(void);
   bool              CreateButton3(void);
   //--- bağımlı olayların işleyicileri
   void              OnClickButton1(void);
   void              OnClickButton2(void);
   void              OnClickButton3(void);
  };
//+------------------------------------------------------------------+
//| Olay İşleme                                                      |
//+------------------------------------------------------------------+
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)
//+------------------------------------------------------------------+
//| Yapıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Yıkıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Oluştur                                                          |
//+------------------------------------------------------------------+
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);
//--- bağımlı kontroller oluştur
   if(!CreateButton1())
      return(false);
   if(!CreateButton2())
      return(false);
   if(!CreateButton3())
      return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| "Button1" düğmesini oluştur                                      |
```

```
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton1(void)
  {
//--- koordinatlar
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+BUTTON_WIDTH;
   int y2=y1+BUTTON_HEIGHT;
//--- oluştur
   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);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| "Button2" düğmesini oluştur                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton2(void)
  {
//--- koordinatlar
   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;
//--- oluştur
   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);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| "Button3" sabit düğmesini oluştur                                |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateButton3(void)
  {
//--- koordinatlar
   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;
//--- oluştur
```

```
   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);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| Olay İşleyici                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton1(void)
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Olay İşleyici                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickButton2(void)
  {
   Comment(__FUNCTION__);
  }
//+------------------------------------------------------------------+
//| Olay İşleyici                                                    |
//+------------------------------------------------------------------+
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()
  {
//--- Uygulama iletişim kutusunu göster
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- uygulamayı çalıştır
   ExtDialog.Run();
//--- başarılı
```

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- yorumları sil
   Comment("");
//--- iletişim kutusunu yok et
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // olay tanıtıcısı
                  const long& lparam,   // long tipli olay parametresi
                  const double& dparam, // double tipli olay parametresi
                  const string& sparam) // string tipli olay parametresi
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Yeni CButton kontrolü oluşturur.

```
virtual bool  Create(
   const long     chart,       // çizelge tanımlayıcısı
   const string   name,        // isim
   const int      subwin,      // çizelge alt-penceresi
   const int      x1,          // x1 koordinatı
   const int      y1,          // y1 koordinatı
   const int      x2,          // x2 koordinatı
   const int      y2           // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Pressed (Get Yöntemi)

Kontrolün basılı olup olmama durumunu ("Pressed" özelliğini) alır.

```
bool  Pressed()  const
```

Dönüş değeri

Kontrol durumu.

Pressed (Set Yöntemi)

Kontrolün basılı olup olmama durumunu ("Pressed" özelliğini) ayarlar.

```
bool  Pressed(
   const bool  pressed      // yeni durum
   )
```

Parametreler

pressed

[in] Yeni kontrol durumu.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Locking (Get Yöntemi)

Kontrolün "Locking" (kilit) özelliğini alır.

```
bool  Locking()  const
```

Dönüş değeri

"Locking" özelliği için yeni değer.

Locking (Set Yöntemi)

Kontrolün "Locking" (kilit) özelliği için yeni değer ayarlar.

```
void  Locking(
   const bool  locking      // yeni değer
   )
```

Parametreler

locking

[in] "Locking" özelliğinin yeni değeri.

Dönüş değeri

Yok.

OnSetText

Kontrolün "SetText" (OBJPROP_TEXT özelliğinin değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnSetText()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetColor

Kontrolün "SetColor" (OBJPROP_COLOR özelliğinin değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnSetColor()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetColorBackground

Kontrolün "SetColorBackground" (OBJPROP_BGCOLOR özelliğinin değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnSetColorBackground()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetColorBorder

Kontrolün "SetColorBorder" (OBJPROP_BORDER_COLOR özelliğinin değişimi) olayının işleyicisi.

```
virtual bool  OnSetColorBackground()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetFont

Kontrolün "SetFont" (OBJPROP_FONT özelliğinin değişimi) olayı için sanal olay işleyici

```
virtual bool  OnSetFont()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetFontSize

Kontrolün "SetFontSize" (OBJPROP_FONTSIZE özelliğinin değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnSetFontSize()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnCreate

Kontrolün "Create" (oluşum) olayı için sanal olay işleyicisi.

```
virtual bool  OnCreate()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnShow

Kotrolün "Show" (göster) olayı için sanal olay işleyici.

```
virtual bool  OnShow()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnHide

Kontrolün "Hide" (gizle) olayı için sanal olay işleyici.

```
virtual bool  OnHide()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnMove

Kontrolün "Move" (taşıma) olayı için sanal olay işleyici.

```
virtual bool  OnMove()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnResize

Kontrolün "Resize" (boyutlandırma) olayı için sanal olay işleyici.

```
virtual bool  OnResize()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnMouseDown

Kontrolün "MouseDown" (fare tuşu basılı) olayı için sanal olay işleyici.

```
virtual bool  OnMouseDown()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"MouseDown" olayı, sol fare tuşu kontrol üzerinde basılı iken gerçekleşir.

OnMouseUp

Kontrolün "MouseUp" (fare tuşu serbest) olayı için sanal olay işleyici.

```
virtual bool  OnMouseUp()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"MouseUp" olayı, sol fare tuşu kontrol üzerinde serbest bırakıldığında gerçekleşir.

CEdit

CEdit sınıfı, "Edit" nesnesi temel alınarak oluşturulmuş bir basit kontrol sınıfıdır.

Açıklama

CEdit sınıfı, kullanıcının metin girişi yapabileceği kontrollerin oluşturulması için tasarlanmıştır.

Bildirim

```
class CEdit : public CWndObj
```

Başlık

```
#include <Controls\Edit.mqh>
```

Kalıtım hiyerarşisi

CObject
  CWnd
    CWndObj
      CEdit

Kodun sonucu aşağıda verilmiştir:

Sınıf Yöntemleri

Oluştur	
Create	Kontrolü oluşturur

Oluştur	
Özellikler	
ReadOnly	"Salt Okunur" özelliğinin değerini alır/ayarlar
TextAlign	"Metin Hizalama" özelliğinin değerini alır/ayarlar
Çizelge nesnesi olay işleyicileri	
OnObjectEndEdit	CHARTEVENT_OBJECT_ENDEDIT sanal olay işleyicisi
Özellik değişimlerinin olay işleyicileri	
OnSetText	"SetText" (Metin değişimi) olayı işleyicisi
OnSetColor	"SetColor" (renk değişimi) olayı işleyicisi
OnSetColorBackground	"SetColorBackground" (arka-plan değişimi) olayı işleyicisi
OnSetColorBorder	"SetColorBorder" (kenarlık rengi değişimi) olayı işleyicisi
OnSetFont	"SetFont" (yazı-tipi değişimi) olayı işleyicisi
OnSetFontSize	"SetFontSize" (yazı-tipi boyutu değişimi) olayı işleyicisi
OnSetZOrder	"SetZOrder" (tıklama önceliği) olay işleyicisi
İçsel olay işleyicileri	
OnCreate	"Create" (oluşturma) olayının işleyicisi
OnShow	"Show" (gösterme) olayının işleyicisi
OnHide	"Hide" (gizleme) olayının işleyicisi
OnMove	"Move" (taşıma) olayının işleyicisi
OnResize	"Resize" (boyutlandırma) olayının işleyicisi
OnChange	"Change" (değişim) olayının işleyicisi
OnClick	"Click" (tıklama) olayının işleyicisi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CWndObj

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Edit kontrolü ile panel oluşturma örneği:

```
//+------------------------------------------------------------------+
//|                                                 ControlsEdit.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Kontrol Panelleri ve İletişim Kutuları. Gösterim sınıfı CEdit"
#include <Controls\Dialog.mqh>
#include <Controls\Edit.mqh>
//+------------------------------------------------------------------+
//| tanımlar                                                         |
//+------------------------------------------------------------------+
//--- girintiler ve boşluklar
#define INDENT_LEFT                         (11)      // sol girinti (izin verilen çe
#define INDENT_TOP                          (11)      // üst girinti (izin verilen çe
#define INDENT_RIGHT                        (11)      // sağ girinti (izin verilen çe
#define INDENT_BOTTOM                       (11)      // alt girinti (izin verilen çe
#define CONTROLS_GAP_X                      (5)       // X koordinatıyla boşluk
#define CONTROLS_GAP_Y                      (5)       // Y kordinatıyla boşluk
//--- düğmeler için
#define BUTTON_WIDTH                        (100)     // X koordinatıyla genişlik
#define BUTTON_HEIGHT                       (20)      // Y koordinatıyla genişlik
//--- gösterge alanı için
#define EDIT_HEIGHT                         (20)      // Y koordinatıyla genişlik
//--- grup kontrolleri için
#define GROUP_WIDTH                         (150)     // X koordinatıyla genişlik
#define LIST_HEIGHT                         (179)     // Y koordinatıyla genişlik
#define RADIO_HEIGHT                        (56)      // Y koordinatıyla genişlik
#define CHECK_HEIGHT                        (93)      // Y koordinatıyla genişlik
//+------------------------------------------------------------------+
//| CControlsDialog Sınıfı                                           |
//| Kullanım: Kontroller uygulamasının ana iletişim kutusu           |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CEdit             m_edit;                          // CEdit nesnesi

public:
```

```
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- oluştur
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- çizelge olay işleyicisi

protected:
   //--- bağımlı kontroller oluştur
   bool              CreateEdit(void);
  };
//+------------------------------------------------------------------+
//| Yapıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Yıkıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Oluştur                                                          |
//+------------------------------------------------------------------+
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);
//--- bağımlı kontroller oluştur
   if(!CreateEdit())
      return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| Görüntü alanını oluştur                                          |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateEdit(void)
  {
//--- koordinatlar
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=ClientAreaWidth()-INDENT_RIGHT;
   int y2=y1+EDIT_HEIGHT;
//--- oluştur
   if(!m_edit.Create(m_chart_id,m_name+"Edit",m_subwin,x1,y1,x2,y2))
      return(false);
//--- içeriğin düzenlenmesine izin ver
```

```
   if(!m_edit.ReadOnly(false))
      return(false);
   if(!Add(m_edit))
      return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Uygulama iletişim kutusunu göster
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- uygulamayı çalıştır
   ExtDialog.Run();
//--- başarılı
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- yorumları sil
   Comment("");
//--- iletişim kutusunu yok et
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // olay tanıtıcısı
                  const long& lparam,   // long tipli olay parametresi
                  const double& dparam, // double tipli olay parametresi
                  const string& sparam) // string tipli olay parametresi
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Yeni CEdit kontrolü oluşturur.

```
virtual bool  Create(
   const long     chart,       // çizelge tanımlayıcısı
   const string   name,        // isim
   const int      subwin,      // çizelge alt-penceresi
   const int      x1,          // x1 koordinatı
   const int      y1,          // y1 koordinatı
   const int      x2,          // x2 koordinatı
   const int      y2           // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

ReadOnly (Get Yöntemi)

Kontrolün "ReadOnly" (salt okunur) özelliğinin değerini alır.

```
bool  ReadOnly()
```

Dönüş değeri

"ReadOnly" özelliğinin değeri.

ReadOnly (Set Yöntemi)

Kontrolün "ReadOnly" (salt okunur) özelliğinin değerini ayarlar.

```
bool  ReadOnly(
   const bool  flag      // yeni bayrak değeri
   )
```

Parametreler

flag

[in] "ReadOnly" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

TextAlign (Get Yöntemi)

Kontrolün "TextAlign" özelliğinin (metin hizalama kipi) değerini alır.

```
ENUM_ALIGN_MODE  TextAlign() const
```

Dönüş değeri

Kontrolün "TextAlign" özelliğinin değeri.

TextAlign (Set Yöntemi)

Kontrolün "TextAlign" özelliğinin (metin hizalama kipi) değerini ayarlar.

```
bool  TextAlign(
   ENUM_ALIGN_MODE  align      // yeni değer
   )
```

Parametreler

align

[in] "Metin Hizalama" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', özellik değişmediyse 'false'.

OnObjectEndEdit

The CHARTEVENT_OBJECT_ENDEDIT event handler.

```
virtual bool  OnObjectEndEdit()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetText

Kontrolün "SetText" (OBJPROP_TEXT özelliğinin değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnSetText()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetColor

Kontrolün "SetColor" (OBJPROP_COLOR özelliğinin değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnSetColor()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetColorBackground

Kontrolün "SetColorBackground" (OBJPROP_BGCOLOR özelliğinin değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnSetColorBackground()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetColorBorder

Kontrolün, "SetColorBorder" (OBJPROP_BORDER_COLOR özelliğinin değişimi) olayı için sanal olay işleyici

```
virtual bool  OnSetColorBackground()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetFont

Kontrolün "SetFont" (OBJPROP_FONT özelliğinin değişimi) olayı için sanal olay işleyici

```
virtual bool  OnSetFont()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetFontSize

Kontrolün "SetFontSize" (OBJPROP_FONTSIZE özelliğinin değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnSetFontSize()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetZOrder

Kontrolün "SetZOrder" (OBJPROP_ZORDER özelliğinin değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnSetZOrder()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnCreate

Kontrolün "Create" (oluşum) olayı için sanal olay işleyicisi.

```
virtual bool  OnCreate()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnShow

Kotrolün "Show" (göster) olayı için sanal olay işleyici.

```
virtual bool  OnShow()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnHide

Kontrolün "Hide" (gizle) olayı için sanal olay işleyici.

```
virtual bool  OnHide()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnMove

Kontrolün "Move" (taşıma) olayı için sanal olay işleyici.

```
virtual bool  OnMove()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnResize

Kontrolün "Resize" (boyutlandırma) olayı için sanal olay işleyici.

```
virtual bool  OnResize()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnChange

Kontrolün "Change" (değişim) olayı için sanal olay işleyicisi.

```
virtual bool  OnChange()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnClick

Kontrolün "Click" (sol fare tuşu tıklaması) olayı için sanal olay işleyici.

```
virtual bool  OnClick()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

CPanel

CPanel, "Dikdörtgen Etiket" grafik nesnesine dayanan basit kontrol sınıfıdır.

Açıklama

CPanel sınıfı bener fonksiyonlara sahip kontrolleri bir grup içerisinde kombine etmek amacıyla tasarlanmıştır.

Bildirim

```
class CPanel : public CWndObj
```

Başlık

```
#include <Controls\Panel.mqh>
```

Kalıtım hiyerarşisi

CObject
CWnd
CWndObj
CPanel

Kodun sonucu aşağıda verilmiştir:

Sınıf Yöntemleri

Oluştur	
Create	Kontrolü oluşturur

Oluştur	
Çizelge nesnesi özellikleri	
BorderType	Çizelge nesnesinin "BorderType" (kenarlık tipi) özelliğinin değerini alır
Çizelge nesnesi olay işleyicileri	
OnSetText	"SetText" (Metin değişimi) olayı işleyicisi
OnSetColorBackground	"SetColorBackground" (arka-plan değişimi) olayı işleyicisi
OnSetColorBorder	"SetColorBorder" (kenarlık rengi değişimi) olayı işleyicisi
İçsel olay işleyicileri	
OnCreate	"Create" (oluşturma) olayının işleyicisi
OnShow	"Show" (gösterme) olayının işleyicisi
OnHide	"Hide" (gizleme) olayının işleyicisi
OnMove	"Move" (taşıma) olayının işleyicisi
OnResize	"Resize" (boyutlandırma) olayının işleyicisi
OnChange	"Change" (değişim) olayının işleyicisi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Dikdörtgen etiket ile panel oluşturma örneği:

```
//+------------------------------------------------------------------+
//|                                                ControlsPanel.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
```

```
#property version   "1.00"
#property description "Kontrol Panelleri ve İletişim Kutuları. Gösterim sınıfı CPanel'
#include <Controls\Dialog.mqh>
//+------------------------------------------------------------------+
//| tanımlar                                                         |
//+------------------------------------------------------------------+
//--- girintiler ve boşluklar
#define INDENT_LEFT                         (11)      // sol girinti (izin verilen çe
#define INDENT_TOP                          (11)      // üst girinti (izin verilen çe
#define INDENT_RIGHT                        (11)      // sağ girinti (izin verilen çe
#define INDENT_BOTTOM                       (11)      // alt girinti (izin verilen çe
#define CONTROLS_GAP_X                      (5)       // X koordinatıyla boşluk
#define CONTROLS_GAP_Y                      (5)       // Y kordinatıyla boşluk
//--- düğmeler için
#define BUTTON_WIDTH                        (100)     // X koordinatıyla genişlik
#define BUTTON_HEIGHT                       (20)      // Y koordinatıyla genişlik
//--- gösterge alanı için
#define EDIT_HEIGHT                         (20)      // Y koordinatıyla genişlik
//--- grup kontrolleri için
#define GROUP_WIDTH                         (150)     // X koordinatıyla genişlik
#define LIST_HEIGHT                         (179)     // Y koordinatıyla genişlik
#define RADIO_HEIGHT                        (56)      // Y koordinatıyla genişlik
#define CHECK_HEIGHT                        (93)      // Y koordinatıyla genişlik
//+------------------------------------------------------------------+
//| CControlsDialog Sınıfı                                           |
//| Kullanım: Kontroller uygulamasının ana iletişim kutusu           |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- oluştur
   virtual bool      Create(const long chart,const string name,const int subwin,const

protected:
   //--- bağımlı kontroller oluştur
   bool              CreatePanel(void);
  };
//+------------------------------------------------------------------+
//| Yapıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Yıkıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
```

```
  {
  }
//+------------------------------------------------------------------+
//| Oluştur                                                          |
//+------------------------------------------------------------------+
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);
//--- bağımlı kontroller oluştur
   if(!CreatePanel())
      return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| "CPanel" oluştur                                                 |
//+------------------------------------------------------------------+
bool CControlsDialog::CreatePanel(void)
  {
//--- koordinatlar
   int x1=20;
   int y1=20;
   int x2=ExtDialog.Width()/3;
   int y2=ExtDialog.Height()/3;
//--- oluştur
   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);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//---
CPanel   my_white_border;        // CPanel nesnesi
bool     pause=true;             // true - durdur
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
```

```
  {
//---
   EventSetTimer(3);
   pause=true;
//--- Uygulama iletişim kutusunu göster
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- uygulamayı çalıştır
   ExtDialog.Run();
//--- başarılı
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- yorumları sil
   Comment("");
//--- iletişim kutusunu yok et
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // olay tanıtıcısı
                  const long& lparam,   // long tipli olay parametresi
                  const double& dparam, // double tipli olay parametresi
                  const string& sparam) // string tipli olay parametresi
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
//+------------------------------------------------------------------+
//|  Timer                                                           |
//+------------------------------------------------------------------+
void OnTimer()
  {
   pause=!pause;
  }
```

Create

Yeni bir CPanel kontrolü oluşturur.

```
virtual bool  Create(
   const long    chart,      // çizelge tanımlayıcısı
   const string  name,       // isim
   const int     subwin,     // çizelge alt-penceresi
   const int     x1,         // x1 koordinatı
   const int     y1,         // y1 koordinatı
   const int     x2,         // x2 koordinatı
   const int     y2          // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

BorderType (Get Yöntemi)

Çizelge nesnesinin "BorderType" (kenarlık tipi) özelliğinin değerini alır.

```
ENUM_BORDER_TYPE  BorderType()
```

Dönüş değeri

"BorderType" özelliğinin değeri.

BorderType (Set Yöntemi)

Çizelge nesnesinin "BorderType" (kenarlık tipi) özelliğinin değerini ayarlar.

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // değer
   )
```

Parametreler

type

[in] "BorderType" özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnSetText

Kontrolün "SetText" (OBJPROP_TEXT özelliğinin değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnSetText()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetColorBackground

Kontrolün "SetColorBackground" (OBJPROP_BGCOLOR özelliğinin değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnSetColorBackground()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnSetColorBorder

Kontrolün, "SetColorBorder" (OBJPROP_BORDER_COLOR özelliğinin değişimi) olayı için sanal olay işleyici

```
virtual bool  OnSetColorBackground()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnCreate

Kontrolün "Create" (oluşum) olayı için sanal olay işleyicisi.

```
virtual bool  OnCreate()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnShow

Kotrolün "Show" (göster) olayı için sanal olay işleyici.

```
virtual bool  OnShow()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnHide

Kontrolün "Hide" (gizle) olayı için sanal olay işleyici.

```
virtual bool  OnHide()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnMove

Kontrolün "Move" (taşıma) olayı için sanal olay işleyici.

```
virtual bool  OnMove()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnResize

Kontrolün "Resize" (boyutlandırma) olayı için sanal olay işleyici.

```
virtual bool  OnResize()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnChange

Kontrolün "Change" (değişim) olayı için sanal olay işleyicisi.

```
virtual bool  OnChange()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

CPicture

CPicture, "Biteşlem etiketi" grafik nesnesi temelinde oluşturulmuş basit kontrol sınıfıdır.

Açıklama

CPicture sınıfı, basit grafik imgeleri oluşturmak için tasarlanmıştır.

Bildirim

```
class CPicture : public CWndObj
```

Başlık

```
#include <Controls\Picture.mqh>
```

Kalıtım hiyerarşisi

CObject
  CWnd
    CWndObj
      CPicture

Kodun sonucu aşağıda verilmiştir:

Sınıf Yöntemleri

Oluştur	
Create	Kontrolü oluşturur

Oluştur	
Çizelge nesnesi özellikleri	
Border	Çizelge nesnesinin kenarlık kalınlığı değerini alır/ayarlar
BmpName	Kontrolün bmp-dosyasının ismini alır/ayarlar
İçsel olaylar	
OnCreate	"Create" (oluşturma) olayının işleyicisi
OnShow	"Show" (gösterme) olayının işleyicisi
OnHide	"Hide" (gizleme) olayının işleyicisi
OnMove	"Move" (taşıma) olayının işleyicisi
OnChange	"Change" (değişim) olayının işleyicisi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Save, Load, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CWndObj

OnEvent, Text, Text, Color, Color, ColorBackground, ColorBackground, ColorBorder, ColorBorder, Font, Font, FontSize, FontSize, ZOrder, ZOrder

Biteşlem etiketi ile panel oluşturma örneği:

```
//+------------------------------------------------------------------+
//|                                                ControlsPicture.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Kontrol Panelleri ve İletişim Kutuları. Gösterim sınıfı CPictu
#include <Controls\Dialog.mqh>
#include <Controls\Picture.mqh>
//+------------------------------------------------------------------+
//| tanımlar                                                         |
//+------------------------------------------------------------------+
//--- girintiler ve boşluklar
```

```
#define INDENT_LEFT                         (11)      // sol girinti (izin verilen çe
#define INDENT_TOP                          (11)      // üst girinti (izin verilen çe
#define INDENT_RIGHT                        (11)      // sağ girinti (izin verilen çe
#define INDENT_BOTTOM                       (11)      // alt girinti (izin verilen çe
#define CONTROLS_GAP_X                      (5)       // X koordinatıyla boşluk
#define CONTROLS_GAP_Y                      (5)       // Y kordinatıyla boşluk
//--- düğmeler için
#define BUTTON_WIDTH                        (100)     // X koordinatıyla genişlik
#define BUTTON_HEIGHT                       (20)      // Y koordinatıyla genişlik
//--- gösterge alanı için
#define EDIT_HEIGHT                         (20)      // Y koordinatıyla genişlik
//--- grup kontrolleri için
#define GROUP_WIDTH                         (150)     // X koordinatıyla genişlik
#define LIST_HEIGHT                         (179)     // Y koordinatıyla genişlik
#define RADIO_HEIGHT                        (56)      // Y koordinatıyla genişlik
#define CHECK_HEIGHT                        (93)      // Y koordinatıyla genişlik
//+------------------------------------------------------------------+
//| CControlsDialog Sınıfı                                           |
//| Kullanım: Kontroller uygulamasının ana iletişim kutusu           |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CPicture          m_picture;                    // CPicture nesnesi

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- oluştur
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- çizelge olay işleyicisi
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- bağımlı kontroller oluştur
   bool              CreatePicture(void);
   //--- bağımlı olayların işleyicileri
   void              OnClickPicture(void);
  };
//+------------------------------------------------------------------+
//| Olay İşleme                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CLICK,m_picture,OnClickPicture)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Yapıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
```

```
  {
  }
//+------------------------------------------------------------------+
//| Yıkıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Oluştur                                                          |
//+------------------------------------------------------------------+
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);
//--- bağımlı kontroller oluştur
   if(!CreatePicture())
      return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| "Picture" oluştur                                                |
//+------------------------------------------------------------------+
bool CControlsDialog::CreatePicture(void)
  {
//--- koordinatlar
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP+(EDIT_HEIGHT+CONTROLS_GAP_Y);
   int x2=x1+32;
   int y2=y1+32;
//--- oluştur
   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);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| Olay İşleyici                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnClickPicture(void)
  {
   Comment(__FUNCTION__);
  }
```

```
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Uygulama iletişim kutusunu göster
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- uygulamayı çalıştır
   ExtDialog.Run();
//--- başarılı
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- yorumları sil
   Comment("");
//--- iletişim kutusunu yok et
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // olay tanıtıcısı
                  const long& lparam,   // long tipli olay parametresi
                  const double& dparam, // double tipli olay parametresi
                  const string& sparam) // string tipli olay parametresi
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Yeni bir CPicture kontrolü oluşturur.

```
virtual bool  Create(
   const long     chart,       // çizelge tanımlayıcısı
   const string   name,        // isim
   const int      subwin,      // çizelge alt-penceresi
   const int      x1,          // x1 koordinatı
   const int      y1,          // y1 koordinatı
   const int      x2,          // x2 koordinatı
   const int      y2           // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Border (Get Yöntemi)

Kontrolün "Border" (kenarlık kalınlığı) özelliğini alır.

```
int  Border()  const
```

Dönüş değeri

"Border" özelliğinin değeri.

Border (Set Yöntemi)

Kontrolün "Border" (kenarlık kalınlığı) özelliğini ayarlar.

```
bool  Border(
   const int  value      // yeni değer
   )
```

Parametreler

value

[in] "Border özelliği için yeni değer.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

BmpName (Get Yöntemi)

Kontrolün bmp-dosyasının ismini alır.

```
string  BmpName()  const
```

Dönüş değeri

Kontrolün bmp-dosyasının ismi.

BmpName (Set Yöntemi)

Kontrolün bmp-dosyasının ismini ayarlar.

```
bool  BmpName(
   const string  name      // dosya ismi
   )
```

Parametreler

name

[in] Kontrolün bmp-dosyası için yeni isim.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnCreate

Kontrolün "Create" (oluşum) olayı için sanal olay işleyicisi.

```
virtual bool  OnCreate()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnShow

Kotrolün "Show" (göster) olayı için sanal olay işleyici.

```
virtual bool  OnShow()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnHide

Kontrolün "Hide" (gizle) olayı için sanal olay işleyici.

```
virtual bool  OnHide()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnMove

Kontrolün "Move" (taşıma) olayı için sanal olay işleyici.

```
virtual bool  OnMove()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnChange

Kontrolün "Change" (değişim) olayı için sanal olay işleyicisi.

```
virtual bool  OnChange()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

CScroll

CScroll, kaydırma çubuklarının oluşturulması için tasarlanmış bir temel sınıftır.

Açıklama

CScroll sınıfı bir karmaşık kontroldür ve kaydırma çubukları oluşturmaya yarayan temel işlevleri içerir. Temel sınıfın kendisi ayrı bir kontrol olarak kullanılmaz. Bunun yerine, soyundan gelen CScrollV ve CScrollH sınıfları kullanılır.

Bildirim

```
class CScroll : public CWndContainer
```

Başlık

```
#include <Controls\Scrolls.mqh>
```

Kalıtım hiyerarşisi

CObject
  CWnd
    CWndContainer
      CScroll

İlk nesil

CScrollH, CScrollV

Sınıf Yöntemleri

Oluştur	
Create	Kontrolü oluşturur
Çizelge nesnesi olay işleyicileri	
OnEvent	Tüm çizelge olayları için olay işleyicisi
Özellikler	
MinPos	Minimal konum bilgisini alır/ ayarlar
MaxPos	Maksimal konum bilgisini alır/ ayarlar
CurrPos	Mevcut konum bilgisini alır/ ayarlar
Bağımlı kontrollerin oluşturulması	
CreateBack	Arka-plan düğmesi oluşturur
CreateInc	Kaydırma çubuğu için "artırma" düğmesi oluşturur

Oluştur	
CreateDec	Kaydırma çubuğu için "azaltma" düğmesi oluşturur
CreateThumb	Kaydırma çubuğu için başparmak düğmesi (sürüklenebilir kontrol) oluşturur
Bağımlı kontroller için olay işleyicileri	
OnClickInc	"Artırma" düğmesinin olayları için olay işleyici
OnClickDec	"Azaltma" düğmesinin olayları için olay işleyici
İçsel olay işleyicileri	
OnShow	"Create" (oluşturma) olayının işleyicisi
OnHide	"Hide" (gizleme) olayının işleyicisi
OnChangePos	"ChangePosition" (konum değişimi) olayı için olay işleyici
Nesne sürükleme işleyicileri	
OnThumbDragStart	"ThumbDragStart" olay işleyicisi
OnThumbDragProcess	"ThumbDragProcess" olay işleyicisi
OnThumbDragEnd	"ThumbDragEnd" olay işleyicisi
Konum	
CalcPos	Kaydırma çubuğu konumunun koordinatlarını alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

Create

Yeni bir CScroll kontrolü oluşturur.

```
virtual bool  Create(
   const long     chart,       // çizelge tanımlayıcısı
   const string   name,        // isim
   const int      subwin,      // çizelge alt-penceresi
   const int      x1,          // x1 koordinatı
   const int      y1,          // y1 koordinatı
   const int      x2,          // x2 koordinatı
   const int      y2           // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnEvent

Çizelge olayı işleyicisi.

```
virtual bool  OnEvent(
   const int      id,          // tanımlayıcı
   const long&    lparam,      // long tipli olay parametresi
   const double&  dparam,      // double tipli olay parametresi
   const string&  sparam       // string tipli olay parametresi
   )
```

Parametreler

id

[in] Olay tanımlayıcısı.

lparam

[in] Referansla geçirilen long tipli olay parametresi.

dparam

[in] Referansla geçirilen double tipli olay parametresi.

sparam

[in] Referansla geçirilen string tipli olay parametresi.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

MinPos (Get Yöntemi)

CScroll kontrolünün "MinPos" (minimal konum) değerini alır.

```
int  MinPos()  const
```

Dönüş değeri

"MinPos" özelliğinin değeri.

MinPos (Set Yöntemi)

CScroll kontrolünün "MinPos" (minimal konum) değerini ayarlar.

```
void  MinPos(
   const int  value      // yeni değer
   )
```

Parametreler

value

[in] "MinPos" özelliğinin yeni değeri.

Dönüş değeri

Yok.

MaxPos (Get method)

CScroll kontrolünün "MaxPos" (maksimal konum) değerini alır.

```
int  MaxPos()  const
```

Dönüş değeri

"MaxPos" özelliğinin yeni değeri.

MaxPos (Set method)

CScroll kontrolünün "MaxPos" (maksimal konum) değerini ayarlar.

```
void  MaxPos(
   const int  value      // yeni değer
   )
```

Parametreler

value

[in] "MaxPos" özelliğinin yeni değeri.

Dönüş değeri

Yok.

CurrPos (Get method)

CScroll kontrolünün "CurrPos" (mevcut konum) değerini alır.

```
int  CurrPos()  const
```

Dönüş değeri

"CurrPos" özelliği için yeni değer.

CurrPos (Set method)

CScroll kontrolünün "CurrPos" (mevcut konum) değerini ayarlar.

```
void  CurrPos(
   const int  value      // yeni değer
   )
```

Parametreler

value

[in] "CurrPos" özelliğinin yeni değeri.

Dönüş değeri

Yok.

CreateBack

CScroll kontrolü için arka-plan düğmesi oluşturur.

```
virtual bool  CreateBack()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateInc

CScroll kontrolü (kaydırma çubuğu) için "artırma" düğmesi oluşturur.

```
virtual bool  CreateInc()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateDec

CScroll kontrolü (kaydırma çubuğu) için "azaltma" düğmesi oluşturur.

```
virtual bool  CreateDec()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateThumb

Kaydırma çubuğu için başparmak düğmesi (sürüklenebilir) oluşturur.

```
virtual bool  CreateThumb()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnClickInc

Kontrolün "ClickInc" (artır düğmesi üzerinde tıklama) olayı için sanal olay işleyici.

```
virtual bool  OnClickInc()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnClickDec

Kontrolün "ClickDec" (azalt düğmesi üzerinde tıklama) olayı için sanal olay işleyici.

```
virtual bool  OnClickDec()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnShow

Kotrolün "Show" (göster) olayı için sanal olay işleyici.

```
virtual bool  OnShow()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnHide

Kontrolün "Hide" (gizle) olayı için sanal olay işleyici.

```
virtual bool  OnHide()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnChangePos

Kontrolün "ChangePos" (konum değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnChangePos()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnThumbDragStart

Kontrolün "ThumbDragStart" (çubuğun sürüklenmesinin başlaması) olayı için sanal olay işleyici.

```
virtual bool  OnThumbDragStart()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"ThumbDragStart" olayı, sürükleme işleminin başlangıcında oluşur.

OnThumbDragProcess

Kontrolün "ThumbDragProcess" (kaydırma çubuğunun taşınması) olayı için sanal olay işleyici.

```
virtual bool  OnThumbDragProcess(
   const int  x,      // x koordinatı
   const int  y       // y koordinatı
   )
```

Parametreler

x

[in] Fare işaretçisinin mevcut X koordinatı.

y

[in] Fare işaretçisinin mevcut Y koordinatı.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"ThumbDragProcess" olayı, kaydırma çubuğu kontrolünün taşınması ile oluşur.

OnThumbDragEnd

Kontrolün "ThumbDragEnd" (taşıma işleminin sonlanması) olayı için sanal olay işleyici.

```
virtual bool  OnThumbDragEnd()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"ThumbDragEnd" olayı, kaydırma çubuğu kontrolünün (thumb button) taşınması bittiğinde oluşur.

CalcPos

Kaydırma çubuğu konumunun koordinatlarını alır.

```
virtual int  CalcPos(
   const int      coord      // koordinatlar
   )
```

Parametreler

coord

[in] Kaydırma çubuğu konumunun koordinatları.

Dönüş değeri

Kaydırma çubuğu konumu.

CScrollV

CScrollV sınıfı, "Dikey kaydırma çubuğu" karmaşık kontrolünün bir sınıfıdır.

Açıklama

CScrollV sınıfı, dikey kaydırma çubukları oluşturmak için tasarlanmıştır.

Bildirim

```
class CScrollV : public CScroll
```

Başlık

```
#include <Controls\Scrolls.mqh>
```

Kalıtım hiyerarşisi

CObject
CWnd
CWndContainer
CScroll
CScrollV

Kodun sonucu aşağıda verilmiştir:

Sınıf Yöntemleri

Bağımlı kontroller	
CreateInc	Kaydırma çubuğunun artırma düğmesini oluşturur
CreateDec	Kaydırma çubuğunun azaltma düğmesini oluşturur
CreateThumb	Kaydırma çubuğunun başparmak düğmesini (taşınabilir kontrol) oluşturur
İçsel olay işleyicileri	
OnResize	"Resize" (boyutlandırma) olayının işleyicisi
OnChangePos	"ChangePosition" (konum değişimi) olayı için olay işleyici
Sürükleme olayı işleyicileri	
OnThumbDragStart	"ThumbDragStart" olay işleyicisi
OnThumbDragProcess	"ThumbDragProcess" olay işleyicisi
OnThumbDragEnd	"ThumbDragEnd" olay işleyicisi
Konum	
CalcPos	Kaydırma çubuğu konumunun koordinatlarını alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

Sınıftan türetilen yöntemler CScroll

Create, OnEvent, MinPos, MinPos, MaxPos, MaxPos, CurrPos, CurrPos

Dikey kaydırma çubuklu panel oluşturma örneği:

```
//+------------------------------------------------------------------+
//|                                                ControlsScrollV.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property description "Kontrol Panelleri ve İletişim Kutuları. Gösterim sınıfı CScroll
#include <Controls\Dialog.mqh>
#include <Controls\Scrolls.mqh>
//+------------------------------------------------------------------+
//| tanımlar                                                         |
//+------------------------------------------------------------------+
//--- girintiler ve boşluklar
#define INDENT_LEFT                         (11)      // sol girinti (izin verilen çe
#define INDENT_TOP                          (11)      // üst girinti (izin verilen çe
#define INDENT_RIGHT                        (11)      // sağ girinti (izin verilen çe
#define INDENT_BOTTOM                       (11)      // alt girinti (izin verilen çe
#define CONTROLS_GAP_X                      (5)       // X koordinatıyla boşluk
#define CONTROLS_GAP_Y                      (5)       // Y kordinatıyla boşluk
//--- düğmeler için
#define BUTTON_WIDTH                        (100)     // X koordinatıyla genişlik
#define BUTTON_HEIGHT                       (20)      // Y koordinatıyla genişlik
//--- gösterge alanı için
#define EDIT_HEIGHT                         (20)      // Y koordinatıyla genişlik
//--- grup kontrolleri için
#define GROUP_WIDTH                         (150)     // X koordinatıyla genişlik
#define LIST_HEIGHT                         (179)     // Y koordinatıyla genişlik
#define RADIO_HEIGHT                        (56)      // Y koordinatıyla genişlik
#define CHECK_HEIGHT                        (93)      // Y koordinatıyla genişlik
//+------------------------------------------------------------------+
//| CControlsDialog Sınıfı                                           |
//| Kullanım: Kontroller uygulamasının ana iletişim kutusu           |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CScrollV          m_scroll_v;                    // CScrollV nesnesi

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- oluştur
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- çizelge olay işleyicisi
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- bağımlı kontroller oluştur
   bool              CreateScrollV(void);
   //--- bağımlı olayların işleyicileri
   void              OnScrollInc(void);
   void              OnScrollDec(void);
  };
//+------------------------------------------------------------------+
//| Olay İşleme                                                      |
```

```
//+------------------------------------------------------------------+
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)
//+------------------------------------------------------------------+
//| Yapıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Yıkıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Oluştur                                                          |
//+------------------------------------------------------------------+
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);
//--- bağımlı kontroller oluştur
   if(!CreateScrollV())
      return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| CScrollsV nesnesini oluştur                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateScrollV(void)
  {
//--- koordinatlar
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=x1+18;
   int y2=y1+LIST_HEIGHT;
//--- oluştur
   if(!m_scroll_v.Create(m_chart_id,m_name+"ScrollV",m_subwin,x1,y1,x2,y2))
      return(false);
//--- kaydırma çubuğunu ayarla
   m_scroll_v.MinPos(0);
//--- kaydırma çubuğunu ayarla
   m_scroll_v.MaxPos(10);
   if(!Add(m_scroll_v))
      return(false);
```

```
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| Olay İşleyici                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollInc(void)
  {
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Olay İşleyici                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollDec(void)
  {
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Uygulama iletişim kutusunu göster
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- uygulamayı çalıştır
   ExtDialog.Run();
//--- başarılı
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- yorumları sil
   Comment("");
//--- iletişim kutusunu yok et
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // olay tanıtıcısı
```

```
                  const long& lparam,   // long tipli olay parametresi
                  const double& dparam, // double tipli olay parametresi
                  const string& sparam) // string tipli olay parametresi
{
 ExtDialog.ChartEvent(id,lparam,dparam,sparam);
}
```

CreateInc

Kaydırma çubuğu için "artırma" düğmesi oluşturur.

```
virtual bool  CreateInc()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateDec

Kaydırma çubuğu için "azaltma" düğmesi oluşturur.

```
virtual bool  CreateDec()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateThumb

Kaydırma çubuğu için başparmak düğmesi (sürüklenebilir kontrol) oluşturur.

```
virtual bool  CreateThumb()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnResize

Kontrolün "Resize" (boyutlandırma) olayı için sanal olay işleyici.

```
virtual bool  OnResize()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnChangePos

Kontrolün "ChangePos" (konum değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnChangePos()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnThumbDragStart

Kontrolün "ThumbDragStart" (çubuğun sürüklenmesinin başlaması) olayı için sanal olay işleyici.

```
virtual bool  OnThumbDragStart()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"ThumbDragStart" olayı, sürükleme işleminin başlangıcında oluşur.

OnThumbDragProcess

Kontrolün "ThumbDragProcess" (kaydırma çubuğunun taşınması) olayı için sanal olay işleyici.

```
virtual bool  OnThumbDragProcess(
   const int  x,      // x koordinatı
   const int  y       // y koordinatı
   )
```

Parametreler

x

[in] Fare işaretçisinin mevcut X koordinatı.

y

[in] Fare işaretçisinin mevcut Y koordinatı.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"ThumbDragProcess" olayı, kaydırma çubuğu kontrolünün taşınması ile oluşur.

OnThumbDragEnd

Kontrolün "ThumbDragEnd" (taşıma işleminin sonlanması) olayı için sanal olay işleyici.

```
virtual bool  OnThumbDragEnd()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"ThumbDragEnd" olayı, kaydırma çubuğu kontrolünün (thumb button) taşınması bittiğinde oluşur.

CalcPos

Kaydırma çubuğu konumunun koordinatlarını alır.

```
virtual int  CalcPos(
   const int      coord      // koordinatlar
   )
```

Parametreler

coord

[in] Kaydırma çubuğu konumunun koordinatları.

Dönüş değeri

Kaydırma çubuğu konumu.

CScrollH

CScrollH, "Yatay kaydırma çubuğu" karmaşık kontrolünün bir sınıfıdır.

Açıklama

CScrollH sınıfı, yatay kaydırma çubukları oluşturmak için tasarlamıştır.

Bildirim

```
class CScrollH : public CScroll
```

Başlık

```
#include <Controls\Scrolls.mqh>
```

Kalıtım hiyerarşisi

CObject
CWnd
CWndContainer
CScroll
CScrollH

Kodun sonucu aşağıda verilmiştir:

Sınıf Yöntemleri

Bağımlı kontroller	
CreateInc	Kaydırma çubuğunun artırma düğmesini oluşturur
CreateDec	Kaydırma çubuğunun azaltma düğmesini oluşturur
CreateThumb	Kaydırma çubuğunun başparmak düğmesini (taşınabilir kontrol) oluşturur
İçsel olay işleyicileri	
OnResize	"Resize" (boyutlandırma) olayının işleyicisi
OnChangePos	"ChangePosition" (konum değişimi) olayı için olay işleyici
Sürükleme olayı işleyicileri	
OnThumbDragStart	"ThumbDragStart" olay işleyicisi
OnThumbDragProcess	"ThumbDragProcess" olay işleyicisi
OnThumbDragEnd	"ThumbDragEnd" olay işleyicisi
Konum	
CalcPos	Kaydırma çubuğu konumunun koordinatlarını alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

Sınıftan türetilen yöntemler CScroll

Create, OnEvent, MinPos, MinPos, MaxPos, MaxPos, CurrPos, CurrPos

Yatay kaydırma çubuklu panel oluşturma örneği:

```
//+------------------------------------------------------------------+
//|                                                ControlsScrollH.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
```

```
#property description "Kontrol Panelleri ve İletişim Kutuları. Gösterim sınıfı CScroll
#include <Controls\Dialog.mqh>
#include <Controls\Scrolls.mqh>
//+------------------------------------------------------------------+
//| tanımlar                                                         |
//+------------------------------------------------------------------+
//--- girintiler ve boşluklar
#define INDENT_LEFT                         (11)      // sol girinti (izin verilen çe
#define INDENT_TOP                          (11)      // üst girinti (izin verilen çe
#define INDENT_RIGHT                        (11)      // sağ girinti (izin verilen çe
#define INDENT_BOTTOM                       (11)      // alt girinti (izin verilen çe
#define CONTROLS_GAP_X                      (5)       // X koordinatıyla boşluk
#define CONTROLS_GAP_Y                      (5)       // Y kordinatıyla boşluk
//--- düğmeler için
#define BUTTON_WIDTH                        (100)     // X koordinatıyla genişlik
#define BUTTON_HEIGHT                       (20)      // Y koordinatıyla genişlik
//--- gösterge alanı için
#define EDIT_HEIGHT                         (20)      // Y koordinatıyla genişlik
//--- grup kontrolleri için
#define GROUP_WIDTH                         (150)     // X koordinatıyla genişlik
#define LIST_HEIGHT                         (179)     // Y koordinatıyla genişlik
#define RADIO_HEIGHT                        (56)      // Y koordinatıyla genişlik
#define CHECK_HEIGHT                        (93)      // Y koordinatıyla genişlik
//+------------------------------------------------------------------+
//| CControlsDialog Sınıfı                                           |
//| Kullanım: Kontroller uygulamasının ana iletişim kutusu           |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CScrollH          m_scroll_v;                    // CScrollH nesnesi

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- oluştur
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- çizelge olay işleyicisi
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- bağımlı kontroller oluştur
   bool              CreateScrollsH(void);
   //--- bağımlı olayların işleyicileri
   void              OnScrollInc(void);
   void              OnScrollDec(void);
  };
//+------------------------------------------------------------------+
//| Olay İşleme                                                      |
```

```
//+------------------------------------------------------------------+
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)
//+------------------------------------------------------------------+
//| Yapıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Yıkıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Oluştur                                                          |
//+------------------------------------------------------------------+
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);
//--- bağımlı kontroller oluştur
   if(!CreateScrollsH())
      return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| CScrollsH nesnesini oluştur                                      |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateScrollsH(void)
  {
//--- koordinatlar
   int x1=INDENT_LEFT;
   int y1=INDENT_TOP;
   int x2=x1+3*BUTTON_WIDTH;
   int y2=y1+18;
//--- oluştur
   if(!m_scroll_v.Create(m_chart_id,m_name+"ScrollsH",m_subwin,x1,y1,x2,y2))
      return(false);
//--- kaydırma çubuğunu ayarla
   m_scroll_v.MinPos(0);
//--- kaydırma çubuğunu ayarla
   m_scroll_v.MaxPos(10);
   if(!Add(m_scroll_v))
      return(false);
```

```
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| Olay İşleyici                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollInc(void)
  {
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Olay İşleyici                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnScrollDec(void)
  {
   Comment("Position of the scrollbar ",m_scroll_v.CurrPos());
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Uygulama iletişim kutusunu göster
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- uygulamayı çalıştır
   ExtDialog.Run();
//--- başarılı
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- yorumları sil
   Comment("");
//--- iletişim kutusunu yok et
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // olay tanıtıcısı
```

```
                  const long& lparam,   // long tipli olay parametresi
                  const double& dparam, // double tipli olay parametresi
                  const string& sparam) // string tipli olay parametresi
{
 ExtDialog.ChartEvent(id,lparam,dparam,sparam);
}
```

CreateInc

Kaydırma çubuğu için "artırma" düğmesi oluşturur.

```
virtual bool  CreateInc()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateDec

Kaydırma çubuğu için "azaltma" düğmesi oluşturur.

```
virtual bool  CreateDec()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateThumb

Kaydırma çubuğu için başparmak düğmesi (sürüklenebilir kontrol) oluşturur.

```
virtual bool  CreateThumb()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnResize

Kontrolün "Resize" (boyutlandırma) olayı için sanal olay işleyici.

```
virtual bool  OnResize()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnChangePos

Kontrolün "ChangePos" (konum değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnChangePos()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnThumbDragStart

Kontrolün "ThumbDragStart" (çubuğun sürüklenmesinin başlaması) olayı için sanal olay işleyici.

```
virtual bool  OnThumbDragStart()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"ThumbDragStart" olayı, sürükleme işleminin başlangıcında oluşur.

OnThumbDragProcess

Kontrolün "ThumbDragProcess" (kaydırma çubuğunun taşınması) olayı için sanal olay işleyici.

```
virtual bool  OnThumbDragProcess(
   const int  x,      // x koordinatı
   const int  y       // y koordinatı
   )
```

Parametreler

x

[in] Fare işaretçisinin mevcut X koordinatı.

y

[in] Fare işaretçisinin mevcut Y koordinatı.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"ThumbDragProcess" olayı, kaydırma çubuğu kontrolünün taşınması ile oluşur.

OnThumbDragEnd

Kontrolün "ThumbDragEnd" (taşıma işleminin sonlanması) olayı için sanal olay işleyici.

```
virtual bool  OnThumbDragEnd()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"ThumbDragEnd" olayı, kaydırma çubuğu kontrolünün (thumb button) taşınması bittiğinde oluşur.

CalcPos

Kaydırma çubuğu konumunun koordinatlarını alır.

```
virtual int  CalcPos(
   const int      coord      // koordinatlar
   )
```

Parametreler

coord

[in] Kaydırma çubuğu konumunun koordinatları.

Dönüş değeri

Kaydırma çubuğu konumu.

CWndClient

CWndClient, "Kullanıcı alanı" karmaşık kontrolünün bir sınıfıdır. Kaydırma çubuğu alanı oluşturmak için bir temel sınıftır.

Açıklama

CWndClient sınıfı, kaydırma çubukları içeren kullanıcı alanı oluşturmak için fonksiyonlar içerir.

Bildirim

```
class CWndClient : public CWndContainer
```

Başlık

```
#include <Controls\WndClient.mqh>
```

Kalıtım hiyerarşisi

CObject
  CWnd
    CWndContainer
      CWndClient

İlk nesil

CCheckGroup, CListView, CRadioGroup

Sınıf Yöntemleri

Oluştur	
Create	Kontrolü oluşturur
Çizelge olay işleyicisi	
OnEvent	Tüm çizelge olayları için olay işleyicisi
Özellikler	
ColorBackground	Arka plan rengini ayarlar
ColorBorder	Kenarlık rengini ayarlar
BorderType	Kenarlık tipini ayarlar
Ayarlar	
VScrolled	Dikey kaydırma çubuğunun kullanım durumunu gösteren bayrağı alır/ayarlar
HScrolled	Yatay kaydırma çubuğunun kullanım durumunu gösteren bayrağı alır/ayarlar
Bağımlı kontroller	

Oluştur	
CreateBack	Kaydırma çubuğu için arka-plan oluşturur
CreateScrollV	Dikey kaydırma çubuğu oluşturur
CreateScrollH	Yatay kaydırma çubuğu oluşturur
İçsel olay işleyicileri	
OnResize	"Resize" (boyutlandırma) olayının işleyicisi
Bağımlı kontroller için olay işleyicileri	
OnVScrollShow	VScroll bağımlı kontrolünün "OnVScrollShow" (kaydırma çubuğunu göster) olayı için sanal olay işleyici
OnVScrollHide	VScroll bağımlı kontrolünün "OnVScrollHide" (kaydırma çubuğunu gizle) olayı için sanal olay işleyici
OnHScrollShow	HScroll bağımlı kontrolünün "Show" (göster) olayı için sanal olay işleyici
OnHScrollHide	"HScroll bağımlı kontrolünün "Hide" (gizle) olayı için sanal olay işleyici
OnScrollLineDown	VScroll bağımlı kontrolünün "ScrollLineDown" (aşağı kaydırma) olayı için sanal olay işleyici
OnScrollLineUp	VScroll bağımlı kontrolünün "ScrollLineUp" (yukarı kaydırma) olayı için sanal olay işleyici
OnScrollLineLeft	VScroll bağımlı kontrolünün "ScrollLineLeft" (sola kaydırma) olayı için sanal olay işleyici
OnScrollLineRight	VScroll bağımlı kontrolünün "ScrollLineRight" (sağa kaydırma) olayı için sanal olay işleyici
Boyutlandırma	
Rebound	CRect sınıfını kullanarak kontrol için yeni koordinatlar ayarlar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CWndContainer

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide, Save, Load

Create

Yeni bir CWndClient kontrolü oluşturur.

```
virtual bool  Create(
   const long     chart,       // çizelge tanımlayıcısı
   const string   name,        // isim
   const int      subwin,      // çizelge alt-penceresi
   const int      x1,          // x1 koordinatı
   const int      y1,          // y1 koordinatı
   const int      x2,          // x2 koordinatı
   const int      y2           // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnEvent

Çizelge olayı işleyicisi.

```
virtual bool  OnEvent(
   const int      id,          // tanımlayıcı
   const long&    lparam,      // long tipli olay parametresi
   const double&  dparam,      // double tipli olay parametresi
   const string&  sparam       // string tipli olay parametresi
   )
```

Parametreler

id

[in] Olay tanımlayıcısı.

lparam

[in] Referansla geçirilen long tipli olay parametresi.

dparam

[in] Referansla geçirilen double tipli olay parametresi.

sparam

[in] Referansla geçirilen string tipli olay parametresi.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

ColorBackground

Kontrolün arka-plan rengini ayarlar.

```
bool  ColorBackground(
   const color  value      // yeni renk
   )
```

Parametreler

value

[in] Kontrolün yeni arka-plan rengi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

ColorBorder

Kontrolün kenarlık rengini ayarlar.

```
bool  ColorBorder(
   const color  value      // renk
   )
```

Parametreler

value

[in] Kontrolün yeni kenarlık rengi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

BorderType

Kontrolün kenarlık tipini ayarlar.

```
bool  BorderType(
   const ENUM_BORDER_TYPE  type      // kenarlık tipi
   )
```

Parametreler

type

[in] Kontrolün kenarlık tipi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

VScrolled (Get Yöntemi)

Dikey kaydırma çubuğunun kullanım durumunu gösteren bayrağı alır.

```
bool  VScrolled()
```

Dönüş değeri

Dikey kaydırma çubuğunun kullanıyorsa 'true', aksi durumda 'false'.

VScrolled (Set Yöntemi)

Dikey kaydırma çubuğunun kullanım durumunu gösteren bayrağı ayarlar.

```
bool  VScrolled(
   const bool  flag      // bayrak
   )
```

Parametreler

flag

[in] Bayrak.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

HScrolled (Get Yöntemi)

Yatay kaydırma çubuğunun kullanım durumunu gösteren bayrağı alır.

```
bool  HScrolled()
```

Dönüş değeri

Yatay kaydırma çubuğu kullanılıyorsa 'true', aksi durumda 'false'.

HScrolled (Set Yöntemi)

Yatay kaydırma çubuğunun kullanım durumunu gösteren bayrağı ayarlar.

```
bool  HScrolled(
   const bool  flag      // bayrak
   )
```

Parametreler

flag

[in] Bayrak.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateBack

Kontrolün arkaplan düğmesini oluşturur.

```
virtual bool  CreateBack()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateScrollV

Dikey kaydırma çubuğu oluşturur.

```
virtual bool  CreateScrollV()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateScrollH

Yatay kaydırma çubuğu oluşturur.

```
virtual bool  CreateScrollH()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnResize

Kontrolün "Resize" (boyutlandırma) olayı için sanal olay işleyici.

```
virtual bool  OnResize()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnVScrollShow

Kontrolün "VScrollShow" (dikey kaydıma çubuğunu göster) olayı için sanal olay işleyici.

```
virtual bool  OnVScrollShow()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnVScrollHide

Kontrolün "VScrollHide" (dikey kaydıma çubuğunu gizle) olayı için sanal olay işleyici.

```
virtual bool  OnVScrollHide()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnHScrollShow

Kontrolün "HScrollShow" (yatay kaydıma çubuğunu göster) olayı için sanal olay işleyici.

```
virtual bool  OnHScrollShow()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnHScrollHide

Kontrolün "HScrollHide" (dikey kaydıma çubuğunu gizle) olayı için sanal olay işleyici.

```
virtual bool  OnHScrollHide()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnScrollLineDown

Kontrolün "ScrollLineDown" (aşağı kaydır) olayı için sanal olay işleyici.

```
virtual bool  OnScrollLineDown()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnScrollLineUp

Kontrolün "ScrollLineUp" (yukarı kaydır) olayı için sanal olay işleyici.

```
virtual bool  OnScrollLineUp()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnScrollLineLeft

Kontrolün "ScrollLineLeft" (sola kaydırma) olayı için sanal olay işleyici.

```
virtual bool  OnScrollLineLeft()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

OnScrollLineRight

Kontrolün "ScrollLineRight" (sağa kaydırma) olayı için sanal olay işleyici.

```
virtual bool  OnScrollLineRight()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

Temel sınıf yöntemi hiçbir şey yapmaz ve her zaman 'true' dönüşü yapar.

ReBound

CRect sınıfını kullanarak, kontrol için yeni koordinatlar ayarlar.

```
void  ReBound(
   const & CRect  rect      // CRect sınıfı
   )
```

Dönüş değeri

Yok.

CListView

CListView, ListView (liste görünümü) karmaşık kontrolünün sınıfıdır.

Açıklama

CListView sınıfı veri bileşenlerinden oluşan bir liste görüntülemek için tasarlanmıştır.

Bildirim

```
class CListView : public CWndClient
```

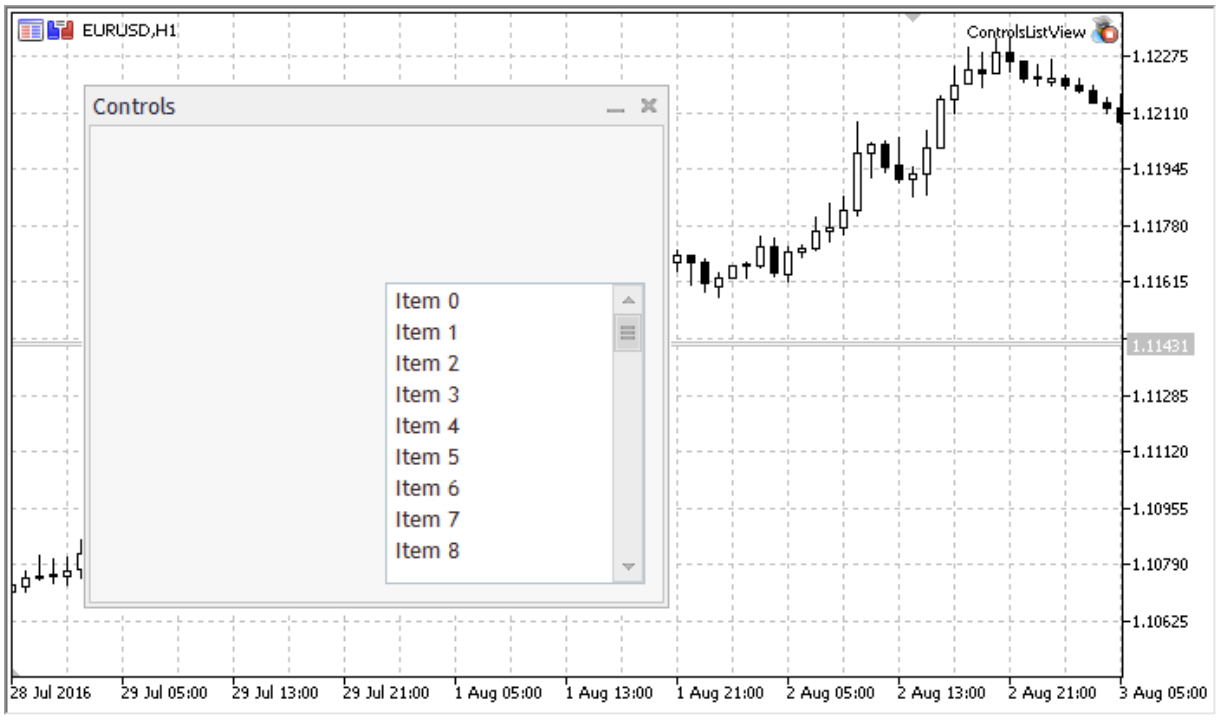
Başlık

```
#include <Controls\ListView.mqh>
```

Kalıtım hiyerarşisi

CObject
CWnd
CWndContainer
CWndClient
CListView

Kodun sonucu aşağıda verilmiştir:



Sınıf Yöntemleri

Oluştur	
Create	Kontrolü oluşturur
Çizelge olay işleyicileri	
OnEvent	Tüm çizelge olayları için olay işleyicisi
Ayarlar	
TotalView	Kontrol üzerinde gösterilen bileşenlerin sayısını ayarlar
Ekle/Sil	
AddItem	Bir bileşen ekler
Veri	
Select	Mevcut listeden indisine göre eleman seçer
SelectByText	Mevcut listeden metnine göre eleman seçer
SelectByValue	Mevcut listeden değerine göre eleman seçer
Salt okunur veri	
Value	Mevcut liste elemanının değerini alır
Bağımlı kontroller	
CreateRow	Bir ListView satırı oluşturur
İçsel olay işleyicileri	
OnResize	"Resize" (yeniden boyutlandırma) olayı için sanal olay işleyicisi
Bağımlı kontroller için olay işleyicileri	
OnVScrollShow	VScroll bağımlı kontrolünün "OnVScrollShow" (kaydırma çubuğunu göster) olayı için sanal olay işleyici
OnVScrollHide	VScroll bağımlı kontrolünün "OnVScrollHide" (kaydırma çubuğunu gizle) olayı için sanal olay işleyici
OnScrollLineDown	VScroll bağımlı kontrolünün "ScrollLineDown" (aşağı kaydırma) olayı için sanal olay işleyici
OnScrollLineUp	VScroll bağımlı kontrolünün "ScrollLineUp" (yukarı kaydırma) olayı için sanal olay işleyici
OnItemClick	"ItemClick" (liste bileşeninin tıklanması) olayı için sanal olay işleyici
Yeniden çizim	
Redraw	Kontrolü yeniden çizer
RowState	Belirtilen satırın durumunu ayarlar
CheckView	Belirtilen satırın görünürlüğünü ayarlar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide, Save, Load

Sınıftan türetilen yöntemler CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Liste görünümü kontrolü ile panel oluşturma örneği:

```
//+------------------------------------------------------------------+
//|                                              ControlsListView.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Kontrol Panelleri ve İletişim Kutuları. Gösterim sınıfı CListVi
#include <Controls\Dialog.mqh>
#include <Controls\ListView.mqh>
//+------------------------------------------------------------------+
//| tanımlar                                                         |
//+------------------------------------------------------------------+
//--- girintiler ve boşluklar
#define INDENT_LEFT                         (11)      // sol girinti (izin verilen çe
#define INDENT_TOP                          (11)      // üst girinti (izin verilen çe
#define INDENT_RIGHT                        (11)      // sağ girinti (izin verilen çe
#define INDENT_BOTTOM                       (11)      // alt girinti (izin verilen çe
#define CONTROLS_GAP_X                      (5)       // X koordinatıyla boşluk
#define CONTROLS_GAP_Y                      (5)       // Y kordinatıyla boşluk
//--- düğmeler için
#define BUTTON_WIDTH                        (100)     // X koordinatıyla genişlik
#define BUTTON_HEIGHT                       (20)      // Y koordinatıyla genişlik
//--- gösterge alanı için
#define EDIT_HEIGHT                         (20)      // Y koordinatıyla genişlik
//--- grup kontrolleri için
#define GROUP_WIDTH                         (150)     // X koordinatıyla genişlik
#define LIST_HEIGHT                         (179)     // Y koordinatıyla genişlik
```

```
#define RADIO_HEIGHT                            (56)      // Y koordinatıyla genişlik
#define CHECK_HEIGHT                            (93)      // Y koordinatıyla genişlik
//+------------------------------------------------------------------+
//| CControlsDialog Sınıfı                                           |
//| Kullanım: Kontroller uygulamasının ana iletişim kutusu           |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CListView         m_list_view;                     // CListView nesnesi

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- oluştur
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- çizelge olay işleyicisi
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- bağımlı kontroller oluştur
   bool              CreateListView(void);
   //--- bağımlı olayların işleyicileri
   void              OnChangeListView(void);
  };
//+------------------------------------------------------------------+
//| Olay İşleme                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_list_view,OnChangeListView)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Yapıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Yıkıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Oluştur                                                          |
//+------------------------------------------------------------------+
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);
//--- bağımlı kontroller oluştur
   if(!CreateListView())
      return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| "ListView" bileşenini oluştur                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateListView(void)
  {
//--- koordinatlar
   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;
//--- oluştur
   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);
//--- dizgilerle doldur
   for(int i=0;i<16;i++)
      if(!m_list_view.AddItem("Item "+IntegerToString(i)))
         return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| Olay İşleyici                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeListView(void)
  {
   Comment(__FUNCTION__+" \""+m_list_view.Select()+"\"");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Uygulama iletişim kutusunu göster
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
```

```
      return(INIT_FAILED);
//--- uygulamayı çalıştır
   ExtDialog.Run();
//--- başarılı
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- yorumları sil
   Comment("");
//--- iletişim kutusunu yok et
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // olay tanıtıcısı
                  const long& lparam,   // long tipli olay parametresi
                  const double& dparam, // double tipli olay parametresi
                  const string& sparam) // string tipli olay parametresi
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

CListView kontrolünü oluşturur.

```
virtual bool  Create(
   const long     chart,       // çizelge tanımlayıcısı
   const string   name,        // isim
   const int      subwin,      // çizelge alt-penceresi
   const int      x1,          // x1 koordinatı
   const int      y1,          // y1 koordinatı
   const int      x2,          // x2 koordinatı
   const int      y2           // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnEvent

Çizelge olayı işleyicisi.

```
virtual bool  OnEvent(
   const int      id,          // tanımlayıcı
   const long&    lparam,      // long tipli olay parametresi
   const double&  dparam,      // double tipli olay parametresi
   const string&  sparam       // string tipli olay parametresi
   )
```

Parametreler

id

[in] Olay tanımlayıcısı.

lparam

[in] Referansla geçirilen long tipli olay parametresi.

dparam

[in] Referansla geçirilen double tipli olay parametresi.

sparam

[in] Referansla geçirilen string tipli olay parametresi.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

TotalView

Kontrol üzerinde gösterilen toplam bileşen sayısını ayarlar.

```
bool  TotalView(
   const int  value      // gösterilecek bileşen sayısı
   )
```

Parametreler

value

[in] Kontrol üzerinde gösterilecek bileşen sayısı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Not

Gösterilen bileşenlerin sayısı sadece bir defaya mahsus olarak ayarlanabilir.

AddItem

Kontrole yeni bir bileşen ekler.

```
bool  AddItem(
   const string  item,      // metin
   const long    value      // değer
   )
```

Parametreler

item

[in] Metin.

value

[in] long tipli değer.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Select

Mevcut listeden indisine göre eleman seçer.

```
bool  Select(
   const int  index     // indis
   )
```

Parametreler

index

[in] Liste elemanının indisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

SelectByText

Mevcut listeden metnine göre eleman seçer.

```
bool  SelectByText(
   const string  text     // metin
   )
```

Parametreler

text

[in] Metin.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

SelectByValue

Mevcut listeden değerine göre eleman seçer.

```
bool  SelectByValue(
   const long  value     // değer
   )
```

Parametreler

value

[in] long tipli değer.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Value

Mevcut liste elemanının değerini alır.

```
long  Value()
```

Dönüş değeri

Mevcut liste elemanının değeri.

CreateRow

CListView kontrolünün bir satırını (liste satırı) oluşturur.

```
bool  CreateRow(
   const int  index     // indis
   )
```

Parametreler

index

[in] Liste elemanının indisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnResize

Kontrolün "Resize" (boyutlandırma) olayı için sanal olay işleyici.

```
virtual bool  OnResize()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnVScrollShow

Kontrolün "VScrollShow" (dikey kaydıma çubuğunu göster) olayı için sanal olay işleyici.

```
virtual bool  OnVScrollShow()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnVScrollHide

Kontrolün "VScrollHide" (dikey kaydıma çubuğunu gizle) olayı için sanal olay işleyici.

```
virtual bool  OnVScrollHide()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnScrollLineDown

Kontrolün "ScrollLineDown" (aşağı kaydır) olayı için sanal olay işleyici.

```
virtual bool  OnScrollLineDown()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnScrollLineUp

Kontrolün "ScrollLineUp" (yukarı kaydır) olayı için sanal olay işleyici.

```
virtual bool  OnScrollLineUp()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnItemClick

"ItemClick" (satır üzerine tıklama) olayının sanal işleyicisi.

```
virtual bool  OnItemClick()
   const int    index     // satır indisi
   )
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Redraw

Kontrolü yeniden çizer.

```
bool  Redraw()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

RowState

Belirtilen liste satırının durumunu ayarlar.

```
bool  RowState(
   const int   index      // indis
   const bool  select     // durum
   )
```

Parametreler

index

[in] Satır indisi.

select

[in] Satır durumu.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CheckView

Belirtilen satırın görünürlüğünü ayarlar.

```
bool  CheckView()
```

Dönüş değeri

Seçilen satır görünür durumdaysa 'true', aksi durumda 'false'.

CComboBox

CComboBox, ComboBox karmaşık kontrolü için tasarlanmış bir sınıftır.

Açıklama

ComboBox, bir statik kontrolle kombine olan, seçim yapmak için tasarlanmış bir liste kutusundan oluşur. Liste-kutusu bölümü, kontrolün yan tarafında bulunan aşağı sürükleme oku kullanılarak aşağı çekilebilir.

Bildirim

```
class CComboBox : public CWndContainer
```

Başlık

```
#include <Controls\ComboBox.mqh>
```

Kalıtım hiyerarşisi

CObject
CWnd
CWndContainer
CComboBox

Kodun sonucu aşağıda verilmiştir:

Sınıf Yöntemleri

Oluştur	
Create	Kontrolü oluşturur
Çizelge olay işleyicileri	
OnEvent	Tüm çizelge olayları için olay işleyicisi
Ekleme	
AddItem	Bir bileşen ekler
Ayarlar	
ListViewItems	Kontrol üzerinde gösterilen bileşenlerin sayısını ayarlar
Veri	
Select	Mevcut listeden indisine göre eleman seçer
SelectByText	Mevcut listeden metnine göre eleman seçer
SelectByValue	Mevcut listeden değerine göre eleman seçer
Salt okunur veri	
Value	Mevcut liste bileşeninin değerini alır
Bağımlı kontroller	
CreateEdit	Bağımlı bir kontrol (düzen) oluşturur
CreateButton	Bağımlı bir kontrol (düğme) oluşturur
CreateList	Bağımlı bir kontrol (liste görünümü) oluşturur
Bağımlı kontroller için olay işleyicileri	
OnClickEdit	"ClickEdit" sanal olay işleyicisi
OnClickButton	"ClickButton" sanal sanal olay işleyicisi
OnChangeList	"ChangeList" sanal olay işleyicisi
Göster/Gizle	
ListShow	Bileşenlerin listesini gösterir
ListHide	Bileşenlerin listesini gizler

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

Sınıftan türetilen yöntemler 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,

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

StateFlagsReset, PropFlags, PropFlags, PropFlagsSet, PropFlagsReset, MouseX, MouseX, MouseY, MouseY, MouseFlags, MouseFlags, MouseFocusKill, BringToTop

Sınıftan türetilen yöntemler CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Hide

Combobox kontrolü ile panel oluşturma örneği:

```
//+------------------------------------------------------------------+
//|                                                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 "Kontrol Panelleri ve İletişim Kutuları. Gösterim sınıfı CComboB
#include <Controls\Dialog.mqh>
#include <Controls\ComboBox.mqh>
//+------------------------------------------------------------------+
//| tanımlar                                                         |
//+------------------------------------------------------------------+
//--- girintiler ve boşluklar
#define INDENT_LEFT                         (11)      // sol girinti (izin verilen çe
#define INDENT_TOP                          (11)      // üst girinti (izin verilen çe
#define INDENT_RIGHT                        (11)      // sağ girinti (izin verilen çe
#define INDENT_BOTTOM                       (11)      // alt girinti (izin verilen çe
#define CONTROLS_GAP_X                      (5)       // X koordinatıyla boşluk
#define CONTROLS_GAP_Y                      (5)       // Y kordinatıyla boşluk
//--- düğmeler için
#define BUTTON_WIDTH                        (100)     // X koordinatıyla genişlik
#define BUTTON_HEIGHT                       (20)      // Y koordinatıyla genişlik
//--- gösterge alanı için
#define EDIT_HEIGHT                         (20)      // Y koordinatıyla genişlik
//--- grup kontrolleri için
#define GROUP_WIDTH                         (150)     // X koordinatıyla genişlik
#define LIST_HEIGHT                         (179)     // Y koordinatıyla genişlik
#define RADIO_HEIGHT                        (56)      // Y koordinatıyla genişlik
#define CHECK_HEIGHT                        (93)      // Y koordinatıyla genişlik
//+------------------------------------------------------------------+
//| CControlsDialog Sınıfı                                           |
//| Kullanım: Kontroller uygulamasının ana iletişim kutusu           |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
```

```
private:
   CComboBox         m_combo_box;;                    // CComboBox nesnesi

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- oluştur
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- çizelge olay işleyicisi
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- bağımlı kontroller oluştur
   bool              CreateComboBox(void);
   //--- bağımlı olayların işleyicileri
   void              OnChangeComboBox(void);
  };
//+------------------------------------------------------------------+
//| Olay İşleme                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_combo_box,OnChangeComboBox)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Yapıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Yıkıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Oluştur                                                          |
//+------------------------------------------------------------------+
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);
//--- bağımlı kontroller oluştur
   if(!CreateComboBox())
      return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
```

```
//| "ComboBox" bileşenini oluştur                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateComboBox(void)
  {
//--- koordinatlar
   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;
//--- oluştur
   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);
//--- dizgilerle doldur
   for(int i=0;i<16;i++)
      if(!m_combo_box.ItemAdd("Item "+IntegerToString(i)))
         return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| Olay İşleyici                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeComboBox(void)
  {
   Comment(__FUNCTION__+" \""+m_combo_box.Select()+"\"");
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Uygulama iletişim kutusunu göster
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- uygulamayı çalıştır
   ExtDialog.Run();
//--- başarılı
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
```

```
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- iletişim kutusunu yok et
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // olay tanıtıcısı
                  const long& lparam,   // long tipli olay parametresi
                  const double& dparam, // double tipli olay parametresi
                  const string& sparam) // string tipli olay parametresi
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Yeni bir CComboBox kontrolü oluşturur.

```
virtual bool  Create(
   const long     chart,       // çizelge tanımlayıcısı
   const string   name,        // isim
   const int      subwin,      // çizelge alt-penceresi
   const int      x1,          // x1 koordinatı
   const int      y1,          // y1 koordinatı
   const int      x2,          // x2 koordinatı
   const int      y2           // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnEvent

Çizelge olayı işleyicisi.

```
virtual bool  OnEvent(
   const int      id,          // tanımlayıcı
   const long&    lparam,      // long tipli olay parametresi
   const double&  dparam,      // double tipli olay parametresi
   const string&  sparam       // string tipli olay parametresi
   )
```

Parametreler

id

[in] Olay tanımlayıcısı.

lparam

[in] Referansla geçirilen long tipli olay parametresi.

dparam

[in] Referansla geçirilen double tipli olay parametresi.

sparam

[in] Referansla geçirilen string tipli olay parametresi.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

AddItem

Kontrole yeni bir bileşen ekler.

```
bool  AddItem(
   const string  item,      // metin
   const long    value      // değer
   )
```

Parametreler

item

[in] Metin.

value=0

[in] long tipli değer.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

ListViewItems

CComboBox kontrolünün liste elemanlarının sayısını ayarlar.

```
void  ListViewItems(
   const int    value     // liste elemanlarının sayısı
   )
```

Parametreler

value

[in] Liste elemanlarının sayısı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Select

Mevcut listeden indisine göre eleman seçer.

```
bool  Select(
   const int  index     // indis
   )
```

Parametreler

index

[in] Liste elemanının indisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

SelectByText

Mevcut listeden metnine göre eleman seçer.

```
bool  SelectByText(
   const string  text     // metin
   )
```

Parametreler

text

[in] Liste elemanının metni.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

SelectByValue

Mevcut listeden değerine göre eleman seçer.

```
bool  SelectByValue(
   const long  value     // değer
   )
```

Parametreler

value

[in] long tipli değer.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Value

Mevcut liste elemanının değerini alır.

```
long  Value()
```

Dönüş değeri

Mevcut liste elemanının değeri.

CreateEdit

Bağımlı bir kontrol (düzenle) oluşturur.

```
virtual bool  CreateEdit()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateButton

Bağımlı bir kontrol (düğme) oluşturur.

```
virtual bool  CreateButton()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateList

Bağımlı bir kontrol (liste görünümü) oluşturur.

```
virtual bool  CreateList()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnClickEdit

kontrolün "ClickEdit" ("düzen" kontrolü üzerinde fare tıklaması) olayı için sanal olay işleyici.

```
virtual bool  OnClickEdit()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnClickButton

Kontrolün "ClickButton" (düğme üzerinde fare tıklaması) olayının -sanal- işleyicisi.

```
virtual bool  OnClickButton()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnChangeList

"ChangeList" (liste değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnChangeList()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

ListShow

Bileşenlerin listesini gösterir.

```
virtual bool  ListShow()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

ListHide

Bileşenler listesini gizler.

```
virtual bool  ListHide()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CCheckBox

Bu sınıf, CheckBox karmaşık kontrolü için tasarlanmıştır.

Açıklama

CCheckBox kontrolü, kullanıcının üstüne tıklaması durumunda daha önceki seçeneği tersine çeviren (iki zıt koşullu) bir onay kutusu görüntüler.

Bildirim

```
class CCheckBox : public CWndContainer
```

Başlık

```
#include <Controls\CheckBox.mqh>
```

Kalıtım hiyerarşisi

CObject
CWnd
CWndContainer
CCheckBox

Kodun sonucu aşağıda verilmiştir:

Sınıf Yöntemleri

Oluşturma	
Create	Kontrolü oluşturur

Oluşturma	
Çizelge olay işleyicileri	
OnEvent	Tüm çizelge olayları için olay işleyicisi
Özellikler	
Metin	Kontrolle ilişkilendirilmiş metin etiketini alır/ayarlar
Color	Kontrolle ilişkilendirilmiş metin etiketinin rengini alır/ayarlar
Durum	
Checked	Kontrolün durum değerini alır/ayarlar
Veri	
Value	Kontrolle ilişkilendirilmiş değeri alır/ayarlar
Bağımlı kontroller	
CreateButton	Bağımlı bir kontrol (düğme) oluşturur
CreateLabel	Bağımlı bir kontrol (etiket) oluşturur
Bağımlı kontroller için olay işleyicileri	
ClickButton	"ClickButton" sanal sanal olay işleyicisi
ClickLabel	"ClickLabel" sanal olay işleyicisi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Checkbox grup kontrolü ile panel oluşturma örneği:

```
//+------------------------------------------------------------------+
//|                                          ControlsCheckGroup.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
```

```
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Kontrol Panelleri ve İletişim Kutuları. Gösterim sınıfı CCheckG
#include <Controls\Dialog.mqh>
#include <Controls\CheckGroup.mqh>
//+------------------------------------------------------------------+
//| tanımlar                                                         |
//+------------------------------------------------------------------+
//--- girintiler ve boşluklar
#define INDENT_LEFT                         (11)      // sol girinti (izin verilen çer
#define INDENT_TOP                          (11)      // üst girinti (izin verilen çer
#define INDENT_RIGHT                        (11)      // sağ girinti (izin verilen çer
#define INDENT_BOTTOM                       (11)      // alt girinti (izin verilen çer
#define CONTROLS_GAP_X                      (5)       // X koordinatıyla boşluk
#define CONTROLS_GAP_Y                      (5)       // Y kordinatıyla boşluk
//--- düğmeler için
#define BUTTON_WIDTH                        (100)     // X koordinatıyla genişlik
#define BUTTON_HEIGHT                       (20)      // Y koordinatıyla genişlik
//--- gösterge alanı için
#define EDIT_HEIGHT                         (20)      // Y koordinatıyla genişlik
//--- grup kontrolleri için
#define GROUP_WIDTH                         (150)     // X koordinatıyla genişlik
#define LIST_HEIGHT                         (179)     // Y koordinatıyla genişlik
#define RADIO_HEIGHT                        (56)      // Y koordinatıyla genişlik
#define CHECK_HEIGHT                        (93)      // Y koordinatıyla genişlik
//+------------------------------------------------------------------+
//| CControlsDialog Sınıfı                                           |
//| Kullanım: Kontroller uygulamasının ana iletişim kutusu           |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CCheckGroup       m_check_group;                  // CCheckGroup nesnesi

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- oluştur
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- çizelge olay işleyicisi
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- bağımlı kontroller oluştur
   bool              CreateCheckGroup(void);
   //--- bağımlı olayların işleyicileri
   void              OnChangeCheckGroup(void);
  };
//+------------------------------------------------------------------+
```

```
//| Olay İşleme                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_check_group,OnChangeCheckGroup)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Yapıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Yıkıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Oluştur                                                          |
//+------------------------------------------------------------------+
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);
//--- bağımlı kontroller oluştur
   if(!CreateCheckGroup())
      return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| "CheckGroup" bileşenini oluştur                                  |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckGroup(void)
  {
//--- koordinatlar
   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;
//--- oluştur
   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);
```

```
//--- dizgilerle doldur
   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()));
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| Olay İşleyici                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckGroup(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_check_group.Value()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Uygulama iletişim kutusunu göster
   if(!ExtDialog.Create(ChartID(),"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- uygulamayı çalıştır
   ExtDialog.Run();
//--- başarılı
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- iletişim kutusunu yok et
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // olay tanıtıcısı
                  const long& lparam,   // long tipli olay parametresi
```

```
                  const double& dparam, // double tipli olay parametresi
                  const string& sparam) // string tipli olay parametresi
{
 ExtDialog.ChartEvent(id,lparam,dparam,sparam);
}
```

Create

Yeni CCheckBox kontrolü oluşturur.

```
virtual bool  Create(
   const long     chart,       // çizelge tanımlayıcısı
   const string   name,        // isim
   const int      subwin,      // çizelge alt-penceresi
   const int      x1,          // x1 koordinatı
   const int      y1,          // y1 koordinatı
   const int      x2,          // x2 koordinatı
   const int      y2           // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnEvent

Çizelge olayı işleyicisi.

```
virtual bool  OnEvent(
   const int      id,          // tanımlayıcı
   const long&    lparam,      // long tipli olay parametresi
   const double&  dparam,      // double tipli olay parametresi
   const string&  sparam       // string tipli olay parametresi
   )
```

Parametreler

id

[in] Olay tanımlayıcısı.

lparam

[in] Referansla geçirilen long tipli olay parametresi.

dparam

[in] Referansla geçirilen double tipli olay parametresi.

sparam

[in] Referansla geçirilen string tipli olay parametresi.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Text (Get Yöntemi)

Kontrolle ilişkilendirilmiş etiketin metnini alır.

```
string  Text()
```

Dönüş değeri

Etiket metni.

Text (Set Yöntemi)

Kontrolle ilişkilendirilmiş etiketin metnini ayarlar.

```
bool  Text(
   const string  value      // metin
   )
```

Parametreler

value

[in] Etiketin yeni metni.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Color (Get Yöntemi)

Kontrolle ilişkilendirilmiş etiketin rengini alır

```
color  Color()  const
```

Dönüş değeri

Etiket rengi.

Color (Set Yöntemi)

Kontrolle ilişkilendirilmiş etiket için yeni renk ayarlar.

```
bool  Color(
   const color  value      // renk
   )
```

Parametreler

value

[in] Yeni etiket rengi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Checked (Get Yöntemi)

Kontrolün durum değerini alır.

```
bool  Checked()  const
```

Dönüş değeri

Kontrol durumu.

Checked (Set Yöntemi)

Kontrolün durum değerini ayarlar.

```
bool  Checked(
   const bool  flag      // durum bayrağı
   )
```

Parametreler

flag

[in] Yeni durum.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Value (Get Yöntemi)

Kontrolle ilişkilendirilmiş değeri alır.

```
int  Value()  const
```

Dönüş değeri

Kontrolle ilişkilendirilmiş değer.

Value (Set Yöntemi)

Kontrolle ilişkilendirilmiş değeri ayarlar.

```
void  Value(
   const int  value      // yeni değer
   )
```

Parametreler

value

[in] yeni değer.

Dönüş değeri

Yok.

CreateButton

Bağımlı bir kontrol (düğme) oluşturur.

```
virtual bool  CreateButton()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateLabel

Bağımlı bir kontrol (etiket) oluşturur.

```
virtual bool  CreateLabel()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnClickButton

Kontrolün "ClickButton" (düğme üzerinde fare tıklaması) olayının -sanal- işleyicisi.

```
virtual bool  OnClickButton()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnClickLabel

Kontrolün "ClickLabel" (etiket üzerinde fare tıklaması) içsel olayının sanal işleyicisi.

```
virtual bool  OnClickLabel()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

CCheckGroup

CCheckGroup, CheckGroup karmaşık kontrolü için tasarlanmış bir sınıftır.

Açıklama

CCheckGroup sınıfı, bayrakları görüntülemek ve düzenlemek için kontroller oluşturulabilmesini sağlar.

Bildirim

```
class CCheckGroup : public CWndClient
```

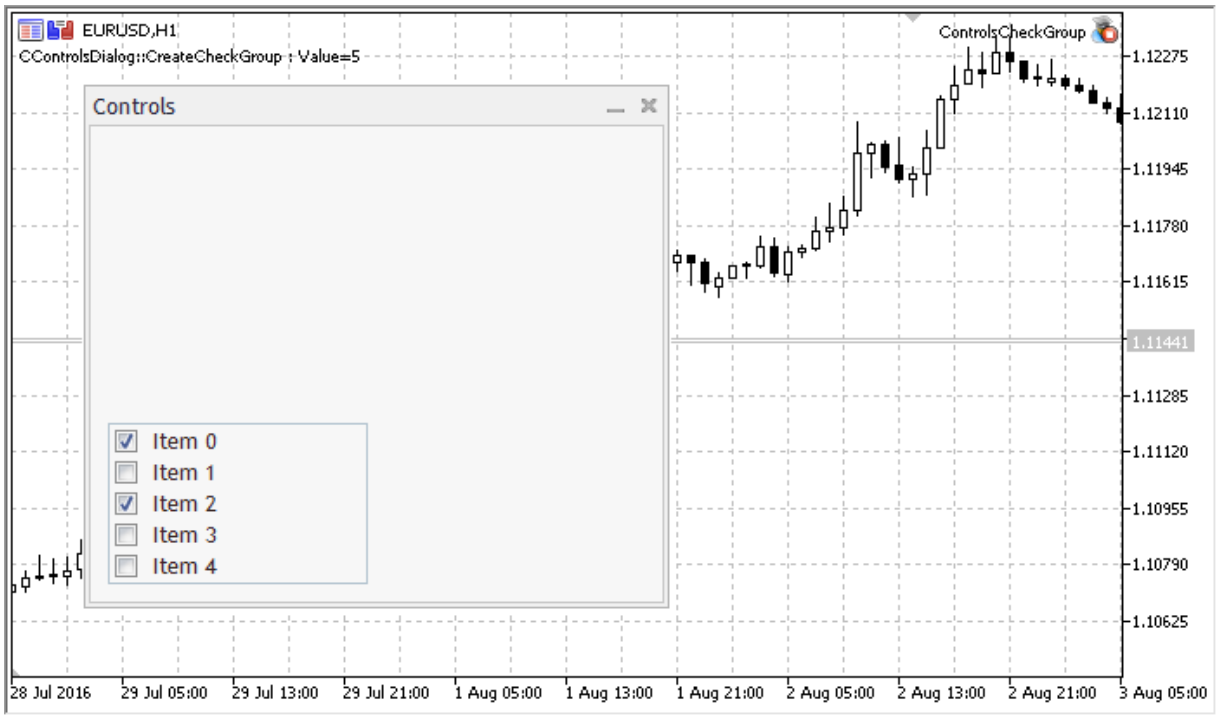
Başlık

```
#include <Controls\CheckGroup.mqh>
```

Kalıtım hiyerarşisi

CObject
  CWnd
    CWndContainer
      CWndClient
        CCheckGroup

Kodun sonucu aşağıda verilmiştir:



Sınıf Yöntemleri

Oluşturma	
Create	Kontrolü oluşturur
Çizelge olay işleyicileri	
OnEvent	Tüm çizelge olayları için olay işleyicisi
Ekleme	
AddItem	Yeni bileşen ekler
Salt okunur veri	
Value	Kontrolle ilişkilendirilen değeri alır
Bağımlı kontroller	
CreateButton	Yeni CCheckBox bileşeni oluşturur
Bağımlı kontroller için olay işleyicileri	
OnVScrollShow	VScroll bağımlı kontrolünün "OnVScrollShow" (kaydırma çubuğunu göster) olayı için sanal olay işleyici
OnVScrollHide	VScroll bağımlı kontrolünün "OnVScrollHide" (kaydırma çubuğunu gizle) olayı için sanal olay işleyici
OnScrollLineDown	VScroll bağımlı kontrolünün "ScrollLineUp" (yukarı kaydırma) olayı için sanal olay işleyici
OnScrollLineUp	VScroll bağımlı kontrolünün "ScrollLineDown" (aşağı kaydırma) olayı için sanal olay işleyici
OnChangeItem	VScroll bağımlı kontrolünün "ChangeItem" (bileşeni değişimi) olayı için sanal olay işleyici
Yeniden çizim	
Redraw	grubu yeniden çizer
RowState	Belirtilen bileşenin durumunu ayarlar

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CWndContainer

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide

Sınıftan türetilen yöntemler CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Checkbox grup kontrolü ile panel oluşturma örneği:

```
//+------------------------------------------------------------------+
//|                                          ControlsCheckGroup.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Kontrol Panelleri ve İletişim Kutuları. Gösterim sınıfı CCheckG
#include <Controls\Dialog.mqh>
#include <Controls\CheckGroup.mqh>
//+------------------------------------------------------------------+
//| tanımlar                                                         |
//+------------------------------------------------------------------+
//--- girintiler ve boşluklar
#define INDENT_LEFT                         (11)      // sol girinti (izin verilen çer
#define INDENT_TOP                          (11)      // üst girinti (izin verilen çer
#define INDENT_RIGHT                        (11)      // sağ girinti (izin verilen çer
#define INDENT_BOTTOM                       (11)      // alt girinti (izin verilen çer
#define CONTROLS_GAP_X                      (5)       // X koordinatıyla boşluk
#define CONTROLS_GAP_Y                      (5)       // Y kordinatıyla boşluk
//--- düğmeler için
#define BUTTON_WIDTH                        (100)     // X koordinatıyla genişlik
#define BUTTON_HEIGHT                       (20)      // Y koordinatıyla genişlik
//--- gösterge alanı için
#define EDIT_HEIGHT                         (20)      // Y koordinatıyla genişlik
//--- grup kontrolleri için
#define GROUP_WIDTH                         (150)     // X koordinatıyla genişlik
#define LIST_HEIGHT                         (179)     // Y koordinatıyla genişlik
#define RADIO_HEIGHT                        (56)      // Y koordinatıyla genişlik
#define CHECK_HEIGHT                        (93)      // Y koordinatıyla genişlik
//+------------------------------------------------------------------+
//| CControlsDialog Sınıfı                                           |
//| Kullanım: Kontroller uygulamasının ana iletişim kutusu           |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
```

```
   CCheckGroup       m_check_group;                    // CCheckGroup nesnesi

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- oluştur
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- çizelge olay işleyicisi
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- bağımlı kontroller oluştur
   bool              CreateCheckGroup(void);
   //--- bağımlı olayların işleyicileri
   void              OnChangeCheckGroup(void);
  };
//+------------------------------------------------------------------+
//| Olay İşleme                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_check_group,OnChangeCheckGroup)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Yapıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Yıkıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Oluştur                                                          |
//+------------------------------------------------------------------+
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);
//--- bağımlı kontroller oluştur
   if(!CreateCheckGroup())
      return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| "CheckGroup" bileşenini oluştur                                  |
```

```
//+------------------------------------------------------------------+
bool CControlsDialog::CreateCheckGroup(void)
  {
//--- koordinatlar
   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;
//--- oluştur
   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);
//--- dizgilerle doldur
   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()));
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| Olay İşleyici                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeCheckGroup(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_check_group.Value()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Uygulama iletişim kutusunu göster
   if(!ExtDialog.Create(ChartID(),"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- uygulamayı çalıştır
   ExtDialog.Run();
//--- başarılı
```

```
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//---
   Comment("");
//--- iletişim kutusunu yok et
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // olay tanıtıcısı
                  const long& lparam,   // long tipli olay parametresi
                  const double& dparam, // double tipli olay parametresi
                  const string& sparam) // string tipli olay parametresi
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Yeni bir CCheckGroup kontrolü oluşturur.

```
virtual bool  Create(
   const long     chart,       // çizelge tanımlayıcısı
   const string   name,        // isim
   const int      subwin,      // çizelge alt-penceresi
   const int      x1,          // x1 koordinatı
   const int      y1,          // y1 koordinatı
   const int      x2,          // x2 koordinatı
   const int      y2           // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnEvent

Çizelge olayı işleyicisi.

```
virtual bool  OnEvent(
   const int      id,          // tanımlayıcı
   const long&    lparam,      // long tipli olay parametresi
   const double&  dparam,      // double tipli olay parametresi
   const string&  sparam       // string tipli olay parametresi
   )
```

Parametreler

id

[in] Olay tanımlayıcısı.

lparam

[in] Referansla geçirilen long tipli olay parametresi.

dparam

[in] Referansla geçirilen double tipli olay parametresi.

sparam

[in] Referansla geçirilen string tipli olay parametresi.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

AddItem

Kontrole yeni bir bileşen ekler.

```
bool  AddItem(
   const string  item,      // metin
   const long    value      // değer
   )
```

Parametreler

item

[in] Metin.

value=0

[in] long tipli değer.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Value

Kontrolle ilişkilendirilmiş değeri alır.

```
long  Value()
```

Dönüş değeri

Kontrolle ilişkilendirilmiş değer.

Not

Değer, CCheckGroup içindeki tüm bileşenlerin durumuna bağlıdır.

CreateButton

Belirtilen indisteki grup içerisinde yeni bir CCheckBox bileşeni oluşturur.

```
bool  CreateButton(
   int  index      // indis
   )
```

Parametreler

index

[in] CCheckGroup içindeki yeni bileşenin indisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnVScrollShow

Kontrolün "VScrollShow" (dikey kaydıma çubuğunu göster) olayı için sanal olay işleyici.

```
virtual bool  OnVScrollShow()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnVScrollHide

Kontrolün "VScrollHide" (dikey kaydıma çubuğunu gizle) olayı için sanal olay işleyici.

```
virtual bool  OnVScrollHide()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnScrollLineDown

Kontrolün "ScrollLineDown" (aşağı kaydır) olayı için sanal olay işleyici.

```
virtual bool  OnScrollLineDown()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnScrollLineUp

Kontrolün "ScrollLineUp" (yukarı kaydır) olayı için sanal olay işleyici.

```
virtual bool  OnScrollLineUp()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnChangeItem

Kontrolün "ChangeItem" (bileşen değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnChangeItem(
   const int  index      // indis
   )
```

Parametreler

index

[in] Değiştirilen bileşenin indisi.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Redraw

Kontrolü yeniden çizer.

```
bool  Redraw()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

RowState

Belirtilen bileşenin durumunu ayarlar.

```
bool  RowState(
   const int   index,      // bileşen indisi
   const bool  select      // yeni durum
   )
```

Parametreler

index

[in] Değiştirilecek bileşenin indisi.

select

[in] Yeni durum.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CRadioButton

CRadioButton sınıfı, RadioButton (radyo düğmesi) karmaşık kontrolü için tasarlanmıştır.

Açıklama

CRadioButton sınıfının kendisi kullanılmaz, CRadioGroup bileşenlerinin oluşturulması için kullanılır.

Bildirim

```
class CRadioButton : public CWndContainer
```

Başlık

```
#include <Controls\RadioButton.mqh>
```

Kalıtım hiyerarşisi

CObject
  CWnd
    CWndContainer
      CRadioButton

Sınıf Yöntemleri

Oluştur	
Create	Kontrolü oluşturur
Çizelge olay işleyicileri	
OnEvent	Tüm çizelge olayları için olay işleyicisi
Özellikler	
Metin	Kontrolle ilişkilendirilmiş metin etiketini alır/ayarlar
Color	Kontrolle ilişkilendirilmiş metin etiketinin rengini alır/ayarlar
Durum	
State	Kontrolün durum değerini alır/ayarlar
Bağımlı kontroller	
CreateButton	Düğme oluşturur
CreateLabel	Etiket oluşturur
Bağımlı kontroller için olay işleyicileri	
OnClickButton	"ClickButton" sanal sanal olay işleyicisi
OnClickLabel	"ClickLabel" sanal olay işleyicisi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide, Save, Load

Create

Yeni bir CRadioButton kontrolü oluşturur.

```
virtual bool  Create(
   const long     chart,       // çizelge tanımlayıcısı
   const string   name,        // isim
   const int      subwin,      // çizelge alt-penceresi
   const int      x1,          // x1 koordinatı
   const int      y1,          // y1 koordinatı
   const int      x2,          // x2 koordinatı
   const int      y2           // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnEvent

Çizelge olayı işleyicisi.

```
virtual bool  OnEvent(
   const int      id,          // tanımlayıcı
   const long&    lparam,      // long tipli olay parametresi
   const double&  dparam,      // double tipli olay parametresi
   const string&  sparam       // string tipli olay parametresi
   )
```

Parametreler

id

[in] Olay tanımlayıcısı.

lparam

[in] Referansla geçirilen long tipli olay parametresi.

dparam

[in] Referansla geçirilen double tipli olay parametresi.

sparam

[in] Referansla geçirilen string tipli olay parametresi.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Text (Get Yöntemi)

Kontrolle ilişkilendirilmiş etiketin metnini alır.

```
string  Text()
```

Dönüş değeri

Etiket metni.

Text (Set Yöntemi)

Kontrolle ilişkilendirilmiş etiketin metnini ayarlar.

```
bool  Text(
   const string  value      // metin
   )
```

Parametreler

value

[in] Etiketin yeni metni.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Color (Get Yöntemi)

Kontrolle ilişkilendirilmiş etiketin rengini alır

```
color  Color()  const
```

Dönüş değeri

Etiket rengi.

Color (Set Yöntemi)

Kontrolle ilişkilendirilmiş etiket için yeni renk ayarlar.

```
bool  Color(
   const color  value      // renk
   )
```

Parametreler

value

[in] Yeni etiket rengi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

State (Get Yöntemi)

Düğme durumunu alır.

```
bool  State()  const
```

Dönüş değeri

Düğme durumu (basılı/serbest).

State (Set Yöntemi)

Düğme durumunu ayarlar.

```
bool  State(
   const bool  flag      // bayrak
   )
```

Parametreler

flag

[in] Yeni düğme durumu.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateButton

Düğme oluşturur.

```
virtual bool  CreateButton()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateLabel

Etiket oluşturur.

```
virtual bool  CreateLabel()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnClickButton

"ClickButton" (düğme üzerinde tıklama) olayının sanal işleyicisi.

```
virtual bool  OnClickButton()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnClickLabel

"ClickLabel" (etiket üzerinde tıklama) olayının sanal işleyicisi.

```
virtual bool  OnClickLabel()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

CRadioGroup

CRadioGroup, RadioGroup karmaşık (bağımlı kontroller içeren) kontrolü için tasarlanmış bir sınıftır.

Açıklama

CRadioGroup sınıfı, kullanıcının, diğer CRadioButton kontrolleriyle eşleştirilmiş bir çok seçenek içerisinden birini seçmesini sağlar.

Bildirim

```
class CRadioGroup : public CWndClient
```

Başlık

```
#include <Controls\RadioGroup.mqh>
```

Kalıtım hiyerarşisi

CObject
CWnd
CWndContainer
CWndClient
CRadioGroup

Kodun sonucu aşağıda verilmiştir:

Sınıf Yöntemleri

Oluştur	
Create	Kontrolü oluşturur
Çizelge olay işleyicileri	
OnEvent	Tüm çizelge olayları için olay işleyicisi
Ekle	
AddItem	Yeni bileşen ekler
Salt okunur veri	
Value	Kontrolle ilişkilendirilen değeri alır
Bağımlı kontroller	
CreateButton	Yeni bir CRadioButton bileşeni oluşturur
Bağımlı kontroller için olay işleyicileri	
OnVScrollShow	VScroll bağımlı kontrolünün "OnVScrollShow" (kaydırma çubuğunu göster) olayı için sanal olay işleyici
OnVScrollHide	VScroll bağımlı kontrolünün "OnVScrollHide" (kaydırma çubuğunu gizle) olayı için sanal olay işleyici
OnScrollLineDown	VScroll bağımlı kontrolünün "ScrollLineDown" (aşağı kaydır) olayı için sanal olay işleyici
OnScrollLineUp	VScroll bağımlı kontrolünün "ScrollLineUp" (yukarı kaydırma) olayı için sanal olay işleyici
OnChangeItem	"ChangeItem" olayı için sanal olay işleyici
Yeniden çizim	
Redraw	Grup bileşenlerini yeniden çizer
RowState	Belirtilen bileşenin durumunu ayarlar
Select	Mevcut bileşeni seçer

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CWndContainer

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Enable, Disable, Hide

Sınıftan türetilen yöntemler CWndClient

ColorBackground, ColorBorder, BorderType, VScrolled, VScrolled, HScrolled, HScrolled, Id

Radyo düğmeleriyle bir panel olşturma örneği:

```
//+------------------------------------------------------------------+
//|                                            ControlsRadioGroup.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
//|                                             https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Kontrol Panelleri ve İletişim Kutuları. Gösterim sınıfı CRadioG
#include <Controls\Dialog.mqh>
#include <Controls\RadioGroup.mqh>
//+------------------------------------------------------------------+
//| tanımlar                                                         |
//+------------------------------------------------------------------+
//--- girintiler ve boşluklar
#define INDENT_LEFT                         (11)      // sol girinti (izin verilen çer
#define INDENT_TOP                          (11)      // üst girinti (izin verilen çer
#define INDENT_RIGHT                        (11)      // sağ girinti (izin verilen çer
#define INDENT_BOTTOM                       (11)      // alt girinti (izin verilen çer
#define CONTROLS_GAP_X                      (5)       // X koordinatıyla boşluk
#define CONTROLS_GAP_Y                      (5)       // Y kordinatıyla boşluk
//--- düğmeler için
#define BUTTON_WIDTH                        (100)     // X koordinatıyla genişlik
#define BUTTON_HEIGHT                       (20)      // Y koordinatıyla genişlik
//--- gösterge alanı için
#define EDIT_HEIGHT                         (20)      // Y koordinatıyla genişlik
//--- grup kontrolleri için
#define GROUP_WIDTH                         (150)     // X koordinatıyla genişlik
#define LIST_HEIGHT                         (179)     // Y koordinatıyla genişlik
#define RADIO_HEIGHT                        (56)      // Y koordinatıyla genişlik
#define CHECK_HEIGHT                        (93)      // Y koordinatıyla genişlik
//+------------------------------------------------------------------+
//| CControlsDialog Sınıfı                                           |
//| Kullanım: Kontroller uygulamasının ana iletişim kutusu           |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
```

```
   CRadioGroup       m_radio_group;                    // CRadioGroup nesnesi

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- oluştur
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- çizelge olay işleyicisi
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- bağımlı kontroller oluştur
   bool              CreateRadioGroup(void);
   //--- bağımlı olayların işleyicileri
   void              OnChangeRadioGroup(void);
  };
//+------------------------------------------------------------------+
//| Olay İşleme                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
ON_EVENT(ON_CHANGE,m_radio_group,OnChangeRadioGroup)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Yapıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Yıkıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Oluştur                                                          |
//+------------------------------------------------------------------+
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);
//--- bağımlı kontroller oluştur
   if(!CreateRadioGroup())
      return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| "RadioGroup" bileşenini oluştur                                  |
```

```
//+------------------------------------------------------------------+
bool CControlsDialog::CreateRadioGroup(void)
  {
//--- koordinatlar
   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;
//--- oluştur
   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);
//--- dizgilerle doldur
   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()));
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| Olay İşleyici                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeRadioGroup(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_radio_group.Value()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Uygulama iletişim kutusunu göster
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- uygulamayı çalıştır
   ExtDialog.Run();
//--- başarılı
   return(INIT_SUCCEEDED);
  }
```

```
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- yorumları sil
   Comment("");
//--- iletişim kutusunu yok et
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // olay tanıtıcısı
                  const long& lparam,   // long tipli olay parametresi
                  const double& dparam, // double tipli olay parametresi
                  const string& sparam) // string tipli olay parametresi
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
  }
```

Create

Yeni bir CRadioGroup kontrolü oluşturur.

```
virtual bool  Create(
   const long     chart,       // çizelge tanımlayıcısı
   const string   name,        // isim
   const int      subwin,      // çizelge alt-penceresi
   const int      x1,          // x1 koordinatı
   const int      y1,          // y1 koordinatı
   const int      x2,          // x2 koordinatı
   const int      y2           // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnEvent

Çizelge olayı işleyicisi.

```
virtual bool  OnEvent(
   const int      id,          // tanımlayıcı
   const long&    lparam,      // long tipli olay parametresi
   const double&  dparam,      // double tipli olay parametresi
   const string&  sparam       // string tipli olay parametresi
   )
```

Parametreler

id

[in] Olay tanımlayıcısı.

lparam

[in] Referansla geçirilen long tipli olay parametresi.

dparam

[in] Referansla geçirilen double tipli olay parametresi.

sparam

[in] Referansla geçirilen string tipli olay parametresi.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

AddItem

Kontrole yeni bir bileşen ekler.

```
bool  AddItem(
   const string  item,      // metin
   const long    value=0    // değer
   )
```

Parametreler

item

[in] Metin.

value=0

[in] long tipli değer.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Value

Kontrolle ilişkilendirilmiş değeri alır.

```
long  Value()
```

Dönüş değeri

Kontrolle ilişkilendirilen değer.

Not

Değer, CRadioGroup içindeki tüm CRadioButton bileşenlerinin durumuna bağlıdır.

CreateButton

Belirtilen indis üzerinde yeni bir CRadioButton bileşeni oluşturur.

```
bool  CreateButton(
   const int  index      // indis
   )
```

Parametreler

index

[in] Yeni CRadioGroup bileşeninin indisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnVScrollShow

Kontrolün "VScrollShow" (dikey kaydıma çubuğunu göster) olayı için sanal olay işleyici.

```
virtual bool  OnVScrollShow()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnVScrollHide

Kontrolün "VScrollHide" (dikey kaydıma çubuğunu gizle) olayı için sanal olay işleyici.

```
virtual bool  OnVScrollHide()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnScrollLineDown

Kontrolün "ScrollLineDown" (aşağı kaydır) olayı için sanal olay işleyici.

```
virtual bool  OnScrollLineDown()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnScrollLineUp

Kontrolün "ScrollLineUp" (yukarı kaydır) olayı için sanal olay işleyici.

```
virtual bool  OnScrollLineUp()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnChangeItem

Kontrolün "ChangeItem" (bileşen değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnChangeItem(
   const int  index      // indis
   )
```

Parametreler

index

[in] Değiştirilen bileşenin indisi.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Redraw

Kontrolü yeniden çizer.

```
bool  Redraw()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

RowState

Belirtilen bileşenin durumunu ayarlar.

```
bool  RowState(
   const int   index,      // bileşen indisi
   const bool  select      // yeni durum
   )
```

Parametreler

index

[in] Değiştirilecek bileşenin indisi.

select

[in] Yeni durum.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Select

Mevcut bileşeni indisi ile seçer.

```
void  Select(
   const int  index      // indis
   )
```

Parametreler

index

[in] Seçilecek bileşenin ismi.

Dönüş değeri

Yok.

CSpinEdit

CSpinEdit sınıfı, SpinEdit karmaşık kontrolünün bir sınıfıdır.

Açıklama

CSpinEdit sınıfı, tamsayı tipli değerlerin düzenlenmesine olanak sağlayan kontrollerin oluşturulması için tasarlanmıştır. Üstteki tuşa basıldığında veriyi otomatik olarak artırırken, alttaki tuşa basıldığında da azaltır.

Bildirim

```
class CSpinEdit : public CWndContainer
```

Başlık

```
#include <Controls\SpinEdit.mqh>
```

Kalıtım hiyerarşisi

CObject
CWnd
CWndContainer
CSpinEdit

Kodun sonucu aşağıda verilmiştir:

Sınıf Yöntemleri

Oluştur	
Create	Kontrolü oluşturur
Çizelge olay işleyicileri	
OnEvent	Tüm çizelge olayları için olay işleyicisi
Özellikler	
MinValue	İzin verilen en düşük değeri alır/ayarlar
MaxValue	İzin verilen en yüksek değeri alır/ayarlar
Durum	
Value	Mevcut değeri alır/ayarlar
Bağımlı kontroller	
CreateEdit	Bağımlı bir kontrol (düzen) oluşturur
CreateInc	"Artırma tuşu" bağımlı kontrolünü oluşturur
CreateDec	"Azaltma tuşu" bağımlı kontrolünü oluşturur
Bağımlı kontroller için olay işleyicileri	
OnClickInc	"ClickInc" sanal olay işleyicisi
OnClickDec	"ClickDec" sanal olay işleyicisi
İçsel olay işleyicileri	
OnChangeValue	"ChangeValue" sanal olay işleyicisi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Listeli düzenleme alanı içeren panel oluşturma örneği:

```
//+------------------------------------------------------------------+
//|                                                ControlsSpinEdit.mq5 |
//|                        Copyright 2000-2024, MetaQuotes Ltd. |
```

```
//|                                                 https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright "Copyright 2017, MetaQuotes Software Corp."
#property link      "https://www.mql5.com"
#property version   "1.00"
#property description "Kontrol Panelleri ve İletişim Kutuları. Gösterim sınıfı CSpinEd
#include <Controls\Dialog.mqh>
#include <Controls\SpinEdit.mqh>
//+------------------------------------------------------------------+
//| tanımlar                                                         |
//+------------------------------------------------------------------+
//--- girintiler ve boşluklar
#define INDENT_LEFT                         (11)      // sol girinti (izin verilen çer
#define INDENT_TOP                          (11)      // üst girinti (izin verilen çer
#define INDENT_RIGHT                        (11)      // sağ girinti (izin verilen çer
#define INDENT_BOTTOM                       (11)      // alt girinti (izin verilen çer
#define CONTROLS_GAP_X                      (5)       // X koordinatıyla boşluk
#define CONTROLS_GAP_Y                      (5)       // Y kordinatıyla boşluk
//--- düğmeler için
#define BUTTON_WIDTH                        (100)     // X koordinatıyla genişlik
#define BUTTON_HEIGHT                       (20)      // Y koordinatıyla genişlik
//--- gösterge alanı için
#define EDIT_HEIGHT                         (20)      // Y koordinatıyla genişlik
//--- grup kontrolleri için
#define GROUP_WIDTH                         (150)     // X koordinatıyla genişlik
#define LIST_HEIGHT                         (179)     // Y koordinatıyla genişlik
#define RADIO_HEIGHT                        (56)      // Y koordinatıyla genişlik
#define CHECK_HEIGHT                        (93)      // Y koordinatıyla genişlik
//+------------------------------------------------------------------+
//| CControlsDialog Sınıfı                                           |
//| Kullanım: Kontroller uygulamasının ana iletişim kutusu           |
//+------------------------------------------------------------------+
class CControlsDialog : public CAppDialog
  {
private:
   CSpinEdit         m_spin_edit;                    // CSpinEdit nesnesi

public:
                     CControlsDialog(void);
                    ~CControlsDialog(void);
   //--- oluştur
   virtual bool      Create(const long chart,const string name,const int subwin,const
   //--- çizelge olay işleyicisi
   virtual bool      OnEvent(const int id,const long &lparam,const double &dparam,cons

protected:
   //--- bağımlı kontroller oluştur
   bool              CreateSpinEdit(void);
   //--- bağımlı olayların işleyicileri
```

```
   void              OnChangeSpinEdit(void);
  };
//+------------------------------------------------------------------+
//| Olay İşleme                                                      |
//+------------------------------------------------------------------+
EVENT_MAP_BEGIN(CControlsDialog)
   ON_EVENT(ON_CHANGE,m_spin_edit,OnChangeSpinEdit)
EVENT_MAP_END(CAppDialog)
//+------------------------------------------------------------------+
//| Yapıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Yıkıcı                                                           |
//+------------------------------------------------------------------+
CControlsDialog::~CControlsDialog(void)
  {
  }
//+------------------------------------------------------------------+
//| Oluştur                                                          |
//+------------------------------------------------------------------+
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);
//--- bağımlı kontroller oluştur
   if(!CreateSpinEdit())
      return(false);
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| "SpinEdit" bileşenini oluştur                                    |
//+------------------------------------------------------------------+
bool CControlsDialog::CreateSpinEdit(void)
  {
//--- koordinatlar
   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;
//--- oluştur
   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()));
//--- başarılı
   return(true);
  }
//+------------------------------------------------------------------+
//| Olay İşleyici                                                    |
//+------------------------------------------------------------------+
void CControlsDialog::OnChangeSpinEdit(void)
  {
   Comment(__FUNCTION__+" : Value="+IntegerToString(m_spin_edit.Value()));
  }
//+------------------------------------------------------------------+
//| Global Variables                                                 |
//+------------------------------------------------------------------+
CControlsDialog ExtDialog;
//+------------------------------------------------------------------+
//| Expert initialization function                                   |
//+------------------------------------------------------------------+
int OnInit()
  {
//--- Uygulama iletişim kutusunu göster
   if(!ExtDialog.Create(0,"Controls",0,40,40,380,344))
      return(INIT_FAILED);
//--- uygulamayı çalıştır
   ExtDialog.Run();
//--- başarılı
   return(INIT_SUCCEEDED);
  }
//+------------------------------------------------------------------+
//| Expert deinitialization function                                 |
//+------------------------------------------------------------------+
void OnDeinit(const int reason)
  {
//--- yorumları sil
   Comment("");
//--- iletişim kutusunu yok et
   ExtDialog.Destroy(reason);
  }
//+------------------------------------------------------------------+
//| Expert chart event function                                      |
//+------------------------------------------------------------------+
void OnChartEvent(const int id,         // olay tanıtıcısı
                  const long& lparam,   // long tipli olay parametresi
                  const double& dparam, // double tipli olay parametresi
                  const string& sparam) // string tipli olay parametresi
  {
   ExtDialog.ChartEvent(id,lparam,dparam,sparam);
```

```
}
```

Create

Yeni bir CSpinEdit kontrolü oluşturur.

```
virtual bool  Create(
   const long     chart,      // çizelge tanımlayıcısı
   const string   name,       // isim
   const int      subwin,     // çizelge alt-penceresi
   const int      x1,         // x1 koordinatı
   const int      y1,         // y1 koordinatı
   const int      x2,         // x2 koordinatı
   const int      y2          // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnEvent

Çizelge olayı işleyicisi.

```
virtual bool  OnEvent(
   const int      id,          // tanımlayıcı
   const long&    lparam,      // long tipli olay parametresi
   const double&  dparam,      // double tipli olay parametresi
   const string&  sparam       // string tipli olay parametresi
   )
```

Parametreler

id

[in] Olay tanımlayıcısı.

lparam

[in] Referansla geçirilen long tipli olay parametresi.

dparam

[in] Referansla geçirilen double tipli olay parametresi.

sparam

[in] Referansla geçirilen string tipli olay parametresi.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

MinValue (Get Yöntemi)

Kontrolün "MinValue" (izin verilen en düşük değer) özelliğinin değerini alır.

```
int  MinValue()  const
```

Dönüş değeri

"MinValue" özelliğinin.

MinValue (Set Yöntemi)

Kontrolün "MinValue" (izin verilen en düşük değer) özelliğinin değerini ayarlar.

```
void  MinValue(
   const int  value      // yeni değer
   )
```

Parametreler

value

[in] "MinValue" özelliğinin yeni değeri.

Dönüş değeri

Yok.

MaxValue (Get Yöntemi)

Kontrolün "MaxValue" (izin verilen en yüksek değer) özelliğinin değerini alır.

```
int  MaxValue()  const
```

Dönüş değeri

"MaxValue" özelliğinin değeri.

MaxValue (Set Yöntemi)

Kontrolün "MaxValue" (izin verilen en yüksek değer) özelliğinin değerini ayarlar.

```
void  MaxValue(
   const int  value      // yeni değer
   )
```

Parametreler

value

[in] "MaxValue" özelliğinin yeni değeri.

Dönüş değeri

Yok.

Value (Get Yöntemi)

Kontrolün "Value" (mevcut değer) özelliğini alır.

```
int  Value()  const
```

Dönüş değeri

"Value" özelliği.

Value (Set Yöntemi)

Kontrolün "Value" (mevcut değer) özelliğini ayarlar.

```
void  Value(
   const int  value      // değer
   )
```

Parametreler

value

[in] Yeni "Value" özelliği.

Dönüş değeri

Yok.

CreateEdit

CEdit bağımlı kontrolünü oluşturur.

```
virtual bool  CreateEdit()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateInc

"Artırma tuşu" bağımlı kontrolünü oluşturur.

```
virtual bool  CreateInc()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateDec

"Azaltma tuşu" bağımlı kontrolünü oluşturur.

```
virtual bool  CreateDec()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnClickInc

Kontrolün "ClickInc" (artır düğmesi üzerinde tıklama) olayı için sanal olay işleyici.

```
virtual bool  OnClickInc()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnClickDec

Kontrolün "ClickDec" (azalt düğmesi üzerinde tıklama) olayı için sanal olay işleyici.

```
virtual bool  OnClickDec()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

OnChangeValue

Kontrolün "ChangeValue" (değer değişimi) olayı için sanal olay işleyici.

```
virtual bool  OnChangeValue()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

CDialog

CDialog, "Diyalog" (iletişim paneli) karmaşık kontrolünün sınıfıdır.

Açıklama

CDialog sınıfı, grup içindeki farklı fonksiyonlara sahip kontrolleri kombine etmek için tasarlanmıştır.

Bildirim

```
class CDialog : public CWndContainer
```

Başlık

```
#include <Controls\Dialog.mqh>
```

Kalıtım hiyerarşisi

CObject
CWnd
CWndContainer
CDialog

İlk nesil

CAppDialog

Sınıf Yöntemleri

Oluşturma	
Create	Kontrolü oluşturur
Çizelge olay işleyicileri	
OnEvent	Tüm çizelge olayları için olay işleyicisi
Özellikler	
Caption	"Caption" (başlık) özelliğinin değerini alır/ayarlar
Ekle	
Add	Kullanıcı alanına kontrol ekler.
Bağımlı kontroller	
CreateWhiteBorder	Bağımlı bir kontrol (beyaz kenarlık) oluşturur
CreateBackground	Bağımlı bir kontrol (arka-plan) oluşturur
CreateCaption	Bağımlı bir kontrol (başlık) oluşturur
CreateButtonClose	Bağımlı bir kontrol (kapatma düğmesi) oluşturur
CreateClientArea	Bağımlı bir kontrol (kullanıcı alanı) oluşturur

Oluşturma	
Bağımlı kontroller için olay işleyicileri	
OnClickCaption	"ClickCaption" sanal olay işleyicisi
OnClickButtonClose	"ClickButtonClose" sanal olay işleyicisi
Kullanıcı alanına erişim	
ClientAreaVisible	Kullanıcı alanının görünür olup olmadığına dair bir değer ayarlar
ClientAreaLeft	Kullanıcı alanının sol üst köşesinin X koordinatını alır
ClientAreaTop	Kullanıcı alanının sol üst köşesinin Y koordinatını alır
ClientAreaRight	Kullanıcı alanının sağ alt köşesinin X koordinatını alır
ClientAreaBottom	Kullanıcı alanının sağ alt köşesinin Y koordinatını alır
ClientAreaWidth	Kullanıcı alanının genişliğini alır
ClientAreaHeight	Kullanıcı alanının yüksekliğini alır
Sürükleme olayı işleyicileri	
OnDialogDragStart	"DialogDragStart" sanal olay işleyicisi
OnDialogDragProcess	"DialogDragProcess" sanal olay işleyicisi
OnDialogDragEnd	"DialogDragEnd" sanal olay işleyicisi

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CWndContainer

Destroy, OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Create

Yeni CDialog kontrolü oluşturur.

```
virtual bool  Create(
   const long     chart,       // çizelge tanımlayıcısı
   const string   name,        // isim
   const int      subwin,      // çizelge alt-penceresi
   const int      x1,          // x1 koordinatı
   const int      y1,          // y1 koordinatı
   const int      x2,          // x2 koordinatı
   const int      y2           // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnEvent

Çizelge olayı işleyicisi.

```
virtual bool  OnEvent(
   const int      id,          // tanımlayıcı
   const long&    lparam,      // long tipli olay parametresi
   const double&  dparam,      // double tipli olay parametresi
   const string&  sparam       // string tipli olay parametresi
   )
```

Parametreler

id

[in] Olay tanımlayıcısı.

lparam

[in] Referansla geçirilen long tipli olay parametresi.

dparam

[in] Referansla geçirilen double tipli olay parametresi.

sparam

[in] Referansla geçirilen string tipli olay parametresi.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Caption (Get Yöntemi)

CDialog kontrolünün "Caption" (başlık) özelliğini alır.

```
string  MinValue()  const
```

Dönüş değeri

"Caption" özelliği.

Caption (Set Yöntemi)

CDialog kontrolünün "Caption" (başlık) özelliğini ayarlar.

```
bool  Caption(
   const string  text      // metin
   )
```

Parametreler

text

[in] "Caption" özelliğinin yeni değeri.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Add

Kullanıcı alanına, işaretçi kullanarak bir kontrol ekler.

```
bool  Add(
   CWnd   *control,         // işaretçi
   )
```

Parametreler

control

[in] Kontrolün işaretçisi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Add

Kullanıcı alanına, referans kullanarak bir kontrol ekler.

```
bool  Add(
   CWnd   &control,         // referans
   )
```

Parametreler

control

[in] Kontrolün referansı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateWhiteBorder

Bağımlı bir kontrol (beyaz kenarlık) oluşturur.

```
virtual bool  CreateWhiteBorder()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateBackground

Bağımlı bir kontrol (arka-plan) oluşturur.

```
virtual bool  CreateBackground()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateCaption

Bağımlı bir kontrol (başlık) oluşturur.

```
virtual bool  CreateCaption()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateButtonClose

Bağımlı bir kontrol (kapatma düğmesi) oluşturur

```
virtual bool  CreateButtonClose()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateClientArea

Bağımlı bir kontrol (kullanıcı alanı) oluşturur.

```
virtual bool  CreateClientArea()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnClickCaption

CDialog kontrolünün "ClickCaption" (başlık çubuğunun fare ile tıklanması) olayı için sanal olay işleyici.

```
virtual bool  OnClickCaption()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

OnClickButtonClose

Kontrolün "ClickButtonClose" (kapatma düğmesinin tıklanması) olayı için sanal olay işleyici.

```
virtual bool  OnClickButtonClose()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

ClientAreaVisible

Kullanıcı alanının görünür olup olmadığına dair bir değer ayarlar.

```
bool  ClientAreaVisible(
   const bool  visible      // görünürlük bayrağı
   )
```

Parametreler

visible

[in] Görünürlük bayrağı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

ClientAreaLeft

Kullanıcı alanının sol üst köşesinin X koordinatını alır.

```
int  ClientAreaLeft()
```

Dönüş değeri

Kullanıcı alanının sol üst köşesinin X koordinatı.

ClientAreaTop

Kullanıcı alanının sol üst köşesinin Y koordinatını alır.

```
int  ClientAreaTop()
```

Dönüş değeri

Kullanıcı alanının sol üst köşesinin Y koordinatı.

ClientAreaRight

Kullanıcı alanının sağ alt köşesinin X koordinatını alır.

```
int  ClientAreaTop()
```

Dönüş değeri

Kullanıcı alanının sağ alt köşesinin X koordinatı.

ClientAreaBottom

Kullanıcı alanının sağ alt köşesinin Y koordinatını alır.

```
int  ClientAreaBottom()
```

Dönüş değeri

Kullanıcı alanının sağ alt köşesinin Y koordinatı.

ClientAreaWidth

Kullanıcı alanının genişliğini alır.

```
int  ClientAreaWidth()
```

Dönüş değeri

Kullanıcı alanının genişliği.

ClientAreaHeight

Kullanıcı alanının yüksekliğini alır.

```
int  ClientAreaHeight()
```

Dönüş değeri

Kullanıcı alanının yüksekliği.

OnDialogDragStart

CDialog kontrolünün "DialogDragStart" olayı için sanal olay işleyici.

```
virtual bool  OnDialogDragStart()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"DialogDragStart" olayı sürükleme işlemi başlatıldığında oluşur.

OnDialogDragProcess

CDialog kontrolünün "DialogDragProcess" olayı için sanal olay işleyici.

```
virtual bool  OnDialogDragProcess()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"DialogDragProcess" olayı, kontrol sürüklendiğinde oluşur.

OnDialogDragEnd

Kontrolün "DialogDragEnd" (sürüklemenin bitmesi) olayı için sanal olay işleyici.

```
virtual bool  OnDialogDragEnd()
```

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Not

"DialogDragEnd" olayı sürükleme işlemi sonlandırıldığında oluşur.

CAppDialog

CAppDialog, "Uygulama Diyaloğu" karmaşık kontrolü için bir sınıftır.

Açıklama

CAppDialog sınıfı, farklı fonksiyonlara sahip kontrollerin bir MQL5 programı içinde bir araya getirilmesi için tasarlanmıştır.

Bildirim

```
class CAppDialog : public CDialog
```

Başlık

```
#include <Controls\Dialog.mqh>
```

Kalıtım hiyerarşisi

CObject
  CWnd
    CWndContainer
      CDialog
        CAppDialog

Sınıf Yöntemleri

Oluşturma ve kaldırma	
Create	Kontrolü oluşturur
Destroy	Kontrolü yok eder
Olayların işlenmesi	
OnEvent	Tüm çizelge olayları için olay işleyicisi
Çalıştırma	
Run	Kontrolü çalıştırır
Çizelge olaylarının işlenmesi	
ChartEvent	Tüm çizelge olayları için olay işleyicisi
Ayarlar	
Minimized	Kontrolün küçültülmüş olup olmadığı hakkında bilgi veren bir değer ayarlar
Kaydetme/Yükleme	
IniFileSave	Kontrol durumunu dosyaya kaydeder
IniFileLoad	Kontrol durumunu dosyadan yükler

Oluşturma ve kaldırma	
IniFileName	Yükleme/kaydetme için dosya ismini ayarlar
IniFileExt	Kontrol durumunu kaydetmek/yüklemek için dosya uzantısı ayarlar
Başlatma	
CreateCommon	Genel başlatma yöntemi
CreateExpert	Uzman Danışmanlarla çalışmak için başlatma yöntemi
CreateIndicator	Göstergelerle çalışmak için başlatma yöntemi
Bağımlı kontroller	
CreateButtonMinMax	Bağımlı kontrolleri oluşturur (simge durumuna küçült/büyüt düğmeleri)
Bağımlı kontroller için olay işleyicileri	
OnClickButtonClose	"ClickButtonClose" (kapatma düğmesinin tıklanması) olayı için sanal olay işleyici
OnClickButtonMinMax	"ClickButtonMinMax" (küçült/büyüt düğmelerinin tıklanması) olayı için sanal olay işleyici
Dışsal olaylar	
OnAnotherApplicationClose	Dışsal olaylar için sanal olay işleyici
Yöntemler	
Rebound	CRect sınıfını kullanarak kontrol için yeni koordinatlar ayarlar
Minimize	Kontrolü küçültülmüş (simge durumuna) durum için ayarlar
Maximize	Kontrolü büyütülmüş durum için ayarlar
CreateInstanceId	Kontrol nesnelerinin isimleri için benzersiz tanımlayıcılar oluşturur
ProgramName	MQL5 programının ismini alır
SubwinOff	Kontrol alt-penceresinin Y konumunu alır

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

Sınıftan türetilen yöntemler 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

Sınıftan türetilen yöntemler CObject

Prev, Prev, Next, Next, Type, Compare

Sınıftan türetilen yöntemler CWndContainer

OnMouseEvent, ControlsTotal, Control, ControlFind, MouseFocusKill, Add, Add, Delete, Delete, Move, Move, Shift, Id, Enable, Disable, Show, Hide

Sınıftan türetilen yöntemler CDialog

Caption, Caption, Add, Add

Create

Yeni CAppDialog kontrolü oluşturur.

```
virtual bool  Create(
   const long     chart,       // çizelge tanımlayıcısı
   const string   name,        // isim
   const int      subwin,      // çizelge alt-penceresi
   const int      x1,          // x1 koordinatı
   const int      y1,          // y1 koordinatı
   const int      x2,          // x2 koordinatı
   const int      y2           // y2 koordinatı
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Destroy

CAppDialog kontrol sonlandırma metodu.

```
virtual void  Destroy(
   const int  reason=REASON_PROGRAM    // neden kodu
)
```

Parametreler

reason

[in] Sonlandırma neden kodu. REASON_PROGRAM varsayılan olarak ayarlandı.

Geri dönüş değeri

Yok.

OnEvent

Çizelge olayı işleyicisi.

```
virtual bool  OnEvent(
   const int      id,          // tanımlayıcı
   const long&    lparam,      // long tipli olay parametresi
   const double&  dparam,      // double tipli olay parametresi
   const string&  sparam       // string tipli olay parametresi
   )
```

Parametreler

id

[in] Olay tanımlayıcısı.

lparam

[in] Referansla geçirilen long tipli olay parametresi.

dparam

[in] Referansla geçirilen double tipli olay parametresi.

sparam

[in] Referansla geçirilen string tipli olay parametresi.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Run

Kontrolü çalıştırır.

```
bool  Run()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

ChartEvent

Çizelge olayı işleyicisi.

```
virtual bool  ChartEvent(
   const int      id,          // tanımlayıcı
   const long&    lparam,      // long tipli olay parametresi
   const double&  dparam,      // double tipli olay parametresi
   const string&  sparam       // string tipli olay parametresi
   )
```

Parametreler

id

[in] Olay tanımlayıcısı.

lparam

[in] Referansla geçirilen long tipli olay parametresi.

dparam

[in] Referansla geçirilen double tipli olay parametresi.

sparam

[in] Referansla geçirilen string tipli olay parametresi.

Dönüş değeri

Olay işlenmişse 'true', aksi durumda 'false'.

Minimized

Kontrolün "(Simge durumuna) Küçültülmüş" durum özelliği için değer ayarlar.

```
bool  Minimized(
   const bool  flag      // durum bayrağı
   )
```

Parametreler

flag

[in] Yeni durum.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

IniFileSave

Kontrol durumunu dosyaya kaydeder.

```
void  IniFileSave()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

IniFileLoad

Kontrol durumunu dosyadan yükler.

```
void  IniFileLoad()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

IniFileName

Yükleme/kaydetme için dosya ismini ayarlar.

```
virtual string  IniFileName()  const
```

Dönüş değeri

Kontrol durumunun yüklenmesi/kaydedilmesi için kullanılan dosya ismi.

Not

Dosya ismi, MQL5 programının çalıştırıldığı Uzman Danışmanın/göstergenin ve çalışılan sembolün ismini içerir.

IniFileExt

Kontrol durumunun yükleneceği/kaydedileceği dosya uzantısını ayarlar.

```
virtual string  IniFileExt()  const
```

Dönüş değeri

Kontrol durumunun yüklenmesi/kaydedilmesi için kullanılan dosya uzantısı.

CreateCommon

Genel başlatma yöntemi.

```
bool  CreateCommon(
   const long    chart,      // çizelge tanımlayıcısı
   const string  name,       // isim
   const int     subwin,     // çizelge alt-penceresi
   )
```

Parametreler

chart

[in] Çizelge tanımlayıcısı.

name

[in] Kontrolün benzersiz ismi.

subwin

[in] Çizelge alt-penceresi.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateExpert

Uzman Danışmanlarla çalışmak için başlatma yöntemi.

```
bool  CreateExpert(
   const int      x1,          // x1 koordinatı
   const int      y1,          // y1 koordinatı
   const int      x2,          // x2 koordinatı
   const int      y2           // y2 koordinatı
   )
```

Parametreler

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateIndicator

Göstergelerle çalışmak için başlatma yöntemi.

```
bool  CreateIndicator(
   const int      x1,          // x1 koordinatı
   const int      y1,          // y1 koordinatı
   const int      x2,          // x2 koordinatı
   const int      y2           // y2 koordinatı
   )
```

Parametreler

x1

[in] Sol üst köşenin X koordinatı.

y1

[in] Sol üst köşenin Y koordinatı.

x2

[in] Sağ alt köşenin X koordinatı.

y2

[in] Sağ alt köşenin Y koordinatı.

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateButtonMinMax

Bağımlı kontroller oluşturur (simge durumuna küçült/ekranı kapla düğmeleri).

```
virtual void  CreateButtonMinMax()
```

Dönüş değeri

Yok.

OnClickButtonClose

Kontrolün "ClickButtonClose" (kapatma düğmesi üzerinde fare tıklaması) olayı için sanal olay işleyici.

```
virtual void  OnClickButtonClose()
```

Dönüş değeri

Yok.

OnClickButtonMinMax

Kontrolün "ClickButtonMinMax" (küçült/büyüt düğmeleri üzerinde fare tıklaması) olayı için sanal olay işleyici.

```
virtual void  OnClickButtonClose()
```

Dönüş değeri

Yok.

OnAnotherApplicationClose

Dışsal olaylar için sanal olay işleyici.

```
virtual void  OnAnotherApplicationClose()
```

Dönüş değeri

Yok.

Rebound

CRect sınıfını kullanarak, kontrol için yeni koordinatlar ayarlar.

```
bool  Rebound(
   const & CRect  rect      // CRect sınıfı
   )
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Minimize

Kontrolü küçültülmüş duruma (simge durumuna) getirir.

```
virtual void  Minimize()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

Maximize

Kontrolü büyütülmüş duruma getirir (ekrana sığdır).

```
virtual void  Maximize()
```

Dönüş değeri

Başarılı ise 'true', aksi durumda 'false'.

CreateInstanceId

Kontrol nesnelerinin isimleri için benzersiz bir tanımlayıcı oluşturur.

```
string  CreateInstanceId()
```

Dönüş değeri

Nesne ismi için ön-ek.

ProgramName

MQL5 programının ismini alır.

```
string  ProgramName()
```

Dönüş değeri

MQL5 programını ismi.

SubwinOff

Kontrol alt-penceresinin Y konumunu alır.

```
void  SubwinOff()
```

Dönüş değeri

Yok.

MQL4'ten MQL5'e Taşınma

MQL5, selefi olan ve sayısız göstergenin, betiğin ve Uzman Danışmanın yazıldığı MQL4 programlama dilinin evrimleşmiş halidir. Bu yeni programlama dilinin eskisiyle büyük oranda uyumluluğu olmasına rağmen, iki dil arasında hala bir takım farklar bulunmaktadır. Programların yeni dile aktarılması sırasında, bu farklara dikkat edilmelidir.

Bu bölüm, MQL4 bilen programcılar için, kodların yeni MQL5 diline aktarılmasını kolaylaştırmak amaçlı bilgiler içermektedir

İlk olarak not edilmesi gerekenler:

- Yeni dilde start(), init() ve deinit() fonksiyonları bulunmamaktadır.
- Kullanılacak gösterge tamponlarının sayısı için sınırlama yoktur.
- DLL dosyaları, Uzman Danışmanın (veya herhangi bir MQL5 programının) başlatılmasının hemen ardından yüklenirler.
- Mantıksal koşulların kontrol süresi kısaltılmıştır.
- Bir dizinin limitinin aşılması durumunda, geçerli işlem sonlandırılır (kritik olarak- Bir hata çıktısı ile).
- Operatörlerin önceliği C++ dilindeki gibidir.
- Bu dil, gizli dönüşüme olanak tanır (dizgiden sayıya bile olsa).
- Yerel değişkenler otomatik olarak başlatılmaz (dizgiler hariç).
- Yaygın yerel diziler otomatik olarak silinir.

Özel Fonksiyonlar init, start ve deinit

MQL4 dili, göstergelerde, uzman danışmanlarda veya scriptlerde kullanılabilecek sadece üç adet ön tanımlı fonksiyon içermekteydi (*.mqh ve kütüphane dosyaları hesaba katılmadan). MQL5 dilinde ise bu fonksiyonlara yer verilmemiştir ama benzerleri mevcuttur. Aşağıdaki tabloda fonksiyonların yaklaşık karşılıkları gösterilmiştir.

MQL4	MQL5
init	OnInit
start	OnStart
deinit	OnDeinit

OnInit ve OnDeinit fonksiyonları, MQL4 dilindeki init ve deinit fonksiyonları ile aynı işlevlere sahiptir - MQL5 programının başlatılması sırasında gerçekleştirilmesi gereken kodların yerleştirilmesi için tasarlanmışlardır. Bu fonksiyonları uygun şekilde yeniden isimlendirebilir veya aynı şekilde bırakabilirsiniz, bunun için fonksiyonları karşılık gelen kısımlara yerleştirmelisiniz.

Örnek:

```
void OnInit()
  {
//--- Başlatma fonksiyonunun çağrısı
   init();
  }
```

```
void OnDeinit(const int reason)
  {
//--- Sonlandırma fonksiyonunun çağrısı
   deinit();
//---
  }
```

Start fonksiyonunun yerine, sadece scriptlerde OnStart kullanılır. Uzman Danışmanlarda ve göstergelerde, sırasıyla OnTick ve OnCalculate şeklinde yeniden isimlendirilir. Bir MQL5 programı içinde çalıştırılacak kodlar, bu üç fonksiyonun içerisine yerleştirilmelidir:

MQL5 program	ana fonksiyon
betik	OnStart
gösterge	OnCalculate
Uzman Danışman	OnTick

Gösterge veya script kodu ana fonksiyonu içermiyorsa veya fonksiyonun ismi istenenden farklıysa, bu fonksiyonun çağrısı gerçekleştirilmeyecektir. Yani, bir betiğin kaynak kodu OnStart işleyicisini içermiyorsa, Bu kod Uzman Danışman şeklinde derlenecektir.

Bir gösterge, OnCalculate fonksiyonunu içermiyorsa, göstergenin derlenmesi olanaksızdır.

Öntanımlı Değişkenler

MQL5 dilinde Ask, Bid ve Bars gibi ön tanımlı değişkenler yoktur. Point ve Digits değişkenleri ise oldukça farklı bir yazıma sahiptirler:

MQL4	MQL5
Digits	_Digits
Point	_Point
	_LastError
	_Period
	_Symbol
	_StopFlag
	_UninitReason

Zaman-Serilerine Erişim

MQL5 dilinde Open[], High[], Low[], Close[], Volume[] ve Time[] gibi ön tanımlı zaman-serileri yer almaz. Bir zaman-serisi için gereken derinlik zaman-serilerine erişim fonksiyonları ile ayarlanabilir.

Uzman Danışmanlar

MQL5 dilinde Uzman Danışmanlar, yeni tik olayları için zorunlu fonksiyonlar gerektirmez - OnTick (MQL4 dilinde yeni tik alındığında start fonksiyonu çalıştırılır). MQL5 dilinde Uzman Danışmanlar, çeşitli tipte olaylar için ön tanımlı işleyici fonksiyonlar içerebilirler :

- OnTick – yeni tik olayı;
- OnTimer – zamanlayıcı olayı;
- OnTrade - alım-satım olayı;
- OnChartEvent – klavyeden veya fareden yapılan giriş olayları, grafiksel nesnelerin taşınması olayı, LabelEdit nesnesinde metin düzenlemenin tamamlanması olayı;
- OnBookEvent – Piyasa Derinliği durumunun değişmesi olayı.

Özel Göstergeler

MQL4 dilinde gösterge tamponlarının sayısı sınırlıdır ve 8 adetten fazla olamaz. MQL5 ise böyle bir sınırlama yoktur, bununla birlikte her gösterge tamponunun, terminalde kaplayacağı yer için belirli bir bellek miktarının tahsisini gerektireceği unutulmamalıdır ve bu yeni imkan istismar edilmemelidir.

MQL4 dilinde 6 farklı tipte özel gösterge çizim stili bulunmaktaydı; buna karşın MQL5 dilinde 18 farklı çizim tipi sunulmaktadır. Çizim stillerinin isimleri değişmemiş, fakat göstergelerin grafiksel temsili belirgin bir şekilde değişmiştir.

Ayrıca, gösterge tamponlarında kullanılan indisleme yönü de farklılık göstermektedir. Varsayılan olarak, MQL5 dilinde gösterge tamponları normal diziler gibi tasarlanmıştır. Yani 0 indisli eleman, geçmişteki en eski olandır ve indis değeri arttıkça, en eski verilerden en yenilerine doğru gelinir.

Özel göstergelerle çalışmak için kullanılan fonksiyonlardan sadece SetIndexBuffer fonksiyonu MQL4 dilindeki şekliyle muhafaza edilmiştir. Ama onun da çağrı şekli değiştirilmiştir; artık gösterge tamponuna bağlanan dizinin içinde depolanacak veri tipini belirtmeniz gerekmektedir.

Özel göstergelerin özellikleri de değiştirilmiş ve genişletilmiştir. Zaman-serilerine erişim için yeni fonksiyonlar eklenmiştir, bu yüzden toplam hesaplama algoritması yeniden gözden geçirilmelidir.

Grafiksel Nesneler

MQL5 içinde, grafiksel nesnelerin sayısı belirgin şekilde artırılmıştır. Artık, grafiksel nesneler herhangi bir zaman aralığındaki çizelge üzerinde, zaman ekseninde saniye çözünürlüğü ile konumlandırılabilmektedir - artık, nesnelerin tutturma noktaları, çizelgenin zaman aralığına göre, çubuk açılış zamanına çekilmemektedir

Bundan böyle Ok, Metin ve Etiket nesneleri için, bağlama yöntemleri belirleyebilir; Etiket, Düğme, Çizelge, Bitmap Etiketi ve Düzenleme nesneleri için nesnenin tutturulacağı çizelge köşesini ayarlayabilirsiniz.

MQL5 Fonksiyonlarının Listesi

Alfabetik sıraya göre tüm MQL5 fonksiyonları.

Fonksiyon	Eylem	Bölüm
AccountInfoDouble	Karşılık gelen hesap özelliğinin double tipli değerine dönüş yapar	Hesap Bilgisi
AccountInfoInteger	Karşılık gelen hesap özelliğinin tamsayı tipli değerine dönüş yapar	Hesap Bilgisi
AccountInfoString	Karşılık gelen hesap özelliğinin string tipli değerine dönüş yapar	Hesap Bilgisi
acos	Bir x değişkeninin ark kosinüs değerine, radyan cinsinden dönüş yapar	Matematik Fonksiyonları
Alert	Ayrı bir pencerede bir mesaj gösterir	Yaygın Fonksiyonlar
ArrayBsearch	Searches for a specified value in a multidimensional numeric array sorted ascending	Dizi Fonksiyonları
ArrayCompare	Basit tipli veya karmaşık nesneler içermeyen özel yapıların karşılaştırma sonucunu verir	Dizi Fonksiyonları
ArrayCopy	Bir diziyi başka bir diziye kopyalar	Dizi Fonksiyonları
ArrayFill	Diziyi belirtilen değerle doldurur	Dizi Fonksiyonları
ArrayFree	Tüm dinamik dizilerin tamponunu (arabelleğini) boşaltır ve sıfır boyutunun eleman sayısını 0 yapar	Dizi Fonksiyonları
ArrayGetAsSeries	Dizinin indisleme yönünü kontrol eder	Dizi Fonksiyonları
ArrayInitialize	Nümerik dizinin tüm elemanlarını belirtilen değerle doldurur	Dizi Fonksiyonları
ArrayIsDynamic	Dizinin dinamik olup olmadığını kontrol eder	Dizi Fonksiyonları
ArrayIsSeries	Dizinin bir zaman serisi olup olmadığını kontrol eder	Dizi Fonksiyonları
ArrayMaximum	Çok boyutlu bir sayısal dizinin ilk boyutundaki en büyük elemanı arar	Dizi Fonksiyonları
ArrayMinimum	Çok boyutlu bir sayısal dizinin ilk boyutundaki en küçük elemanı	Dizi Fonksiyonları

Fonksiyon	Eylem	Bölüm
	arar	
ArrayRange	Dizinin belli bir boyutundaki eleman sayısına dönüş yapar	Dizi Fonksiyonları
ArrayResize	Dizinin ilk boyutunun büyüklüğünü yeniden ayarlar	Dizi Fonksiyonları
ArraySetAsSeries	Dizinin indisleme yönünü ayarlar	Dizi Fonksiyonları
ArraySize	Dizideki eleman sayısına dönüş yapar	Dizi Fonksiyonları
ArraySort	Nümerik dizileri ilk boyuta göre sıralar	Dizi Fonksiyonları
ArrayPrint	Basit tipli bir diziyi veya bir yapıyı günlüğe yazar	Dizi Fonksiyonları
ArrayRange	Dizinin belli bir boyutundaki eleman sayısına dönüş yapar	Dizi Fonksiyonları
ArrayResize	Dizinin ilk boyutunun büyüklüğünü yeniden ayarlar	Dizi Fonksiyonları
ArrayInsert	Belirtilen eleman sayısını belirli bir indeksten başlayarak bir kaynak diziden bir alıcı diziye ekler	Dizi Fonksiyonları
ArrayRemove	Belirtilen eleman sayısını, belirtilen indeksten başlayarak diziden kaldırır	Dizi Fonksiyonları
ArrayReverse	Dizideki belirtilen eleman sayısını, belirtilen indeksten başlayarak tersine çevirir	Matematik Fonksiyonları
ArraySetAsSeries	Dizinin indisleme yönünü ayarlar	Matematik Fonksiyonları
ArraySize	Dizideki eleman sayısına dönüş yapar	Zaman Serilerine ve Göstergelere Erişim
ArraySort	Nümerik dizileri ilk boyuta göre sıralar	Zaman Serilerine ve Göstergelere Erişim
ArraySwap	Aynı türdeki iki dinamik dizinin içeriğini değiştirir	Ekonomik Takvim
CalendarEventById	ID'sine göre olay açıklaması elde et	Ekonomik Takvim
CalendarValueById	ID'sine göre olay değeri açıklaması elde et	Ekonomik Takvim

Fonksiyon	Eylem	Bölüm
CalendarCountries	Takvimde mevcut olan ülke adlarının dizisini elde et	Ekonomik Takvim
CalendarEventByCountry	Takvimdeki tüm olayların açıklamalarını belirtilen ülke koduna göre elde et	Ekonomik Takvim
CalendarEventByCurrency	Takvimdeki tüm olayların açıklamalarını belirtilen para birimine göre elde et	Ekonomik Takvim
CalendarValueHistoryByEvent	Belirtilen zaman aralığındaki tüm olayların değer dizisini olay ID'sine göre elde et	Ekonomik Takvim
CalendarValueHistory	Ülkeye ve/veya para birimine göre sıralan belirli bir zaman aralığındaki tüm olaylar için değer dizisini elde et	Ekonomik Takvim
CalendarValueLastByEvent	Belirtilen bir change_id ile takvim veritabanı durumundan bu yana ID'sine göre olay değer dizisini elde et	Ekonomik Takvim
CalendarValueLast	Belirtilen bir change_id ile takvim veritabanı durumundan bu yana ülkeye ve/veya para birimine göre sıralanan tüm etkinlikler için değer dizisini elde et	Ekonomik Takvim
ceil	Nümerik bir değer için en yakın büyük tamsayı değerine dönüş yapar	Matematik Fonksiyonları
CharArrayToString	Bir sembol kodunu (ansi) tek sembollü bir diziye kopyalar	Dönüşüm Fonksiyonları
ChartApplyTemplate	Belirlenmiş bir dosyadan, belirlenmiş bir şablonu çizelgeye uygular	Çizelge İşlemleri
ChartClose	Belirlenen çizelgeyi kapatır	Çizelge İşlemleri
ChartFirst	Müşteri terminalindeki ilk çizelgenin kimliğine dönüş yapar	Çizelge İşlemleri
ChartGetDouble	Belirlenen çizelgenin double tipli özelliğine dönüş yapar	Çizelge İşlemleri
ChartGetInteger	Belirlenen çizelgenin tamsayı değerli özelliğine dönüş yapar	Çizelge İşlemleri

Fonksiyon	Eylem	Bölüm
ChartGetString	Belirlenen çizelgenin string tipli özelliğine dönüş yapar	Çizelge İşlemleri
ChartID	Mevcut çizelgenin kimliğine dönüş yapar	Çizelge İşlemleri
ChartIndicatorAdd	Belirlenen tanıtıcı değer ile bir göstergeyi, belirlenen bir çizelge penceresine ekler	Çizelge İşlemleri
ChartIndicatorDelete	Belirlenen bir çizelge penceresinden belirlenen isimde bir göstergeyi kaldırır	Çizelge İşlemleri
ChartIndicatorGet	Belirlenen çizelge içinde, belirlenen kısa isimli göstergenin tanıtıcı değerine dönüş yapar	Çizelge İşlemleri
ChartIndicatorName	Belirlenen çizelge üzerindeki gösterge listesindeki numarayı kullanarak göstergenin kısa ismine dönüş yapar	Çizelge İşlemleri
ChartIndicatorsTotal	Belirlenen çizelge penceresine uygulanmış tüm göstergelerin sayısına dönüş yapar	Çizelge İşlemleri
ChartNavigate	Belirlenen çizelge üzerinde, belirlenen konuma göre, belirlenen değerde kaydırma gerçekleştirir	Çizelge İşlemleri
ChartNext	Belirlenmiş çizelgenin sonrasındaki, çizelgenin kimliğine dönüş yapar	Çizelge İşlemleri
ChartOpen	Belirlenmiş sembol ve periyot ile yeni bir çizelge açar	Çizelge İşlemleri
CharToString	Bir sembol kodunu, tek karakterlik bir dizgiye dönüştürür	Dönüşüm Fonksiyonları
ChartPeriod	Belirlenen çizelgenin periyot değerine dönüş yapar	Çizelge İşlemleri
ChartPriceOnDropped	Uzman Danışmanın veya betiğin iliştirildiği çizelge noktasının fiyat koordinatına dönüş yapar	Çizelge İşlemleri
ChartRedraw	Belirlenen çizelgeyi yeniden çizim için zorlar	Çizelge İşlemleri
ChartSaveTemplate	Çizelgenin mevcut ayarlarını belirlenmiş bir isimle bir şablona	Çizelge İşlemleri

Fonksiyon	Eylem	Bölüm
	kaydeder	
ChartScreenShot	Çizelgenin mevcut durumunun GIF, PNG veya BMP formatında ekran görüntüsünü alır	Çizelge İşlemleri
ChartSetDouble	Belirlenen çizelgenin karşılık gelen özelliği için double değer ayarlar	Çizelge İşlemleri
ChartSetInteger	Belirlenen çizelgenin karşılık gelen özelliği için tamsayı (datetime, int, color, bool or char) değer ayarlar	Çizelge İşlemleri
ChartSetString	Belirlenen çizelgenin karşılık gelen özelliği için string tipli değer ayarlar	Çizelge İşlemleri
ChartSetSymbolPeriod	Belirlenen nesnenin sembol değerini ve periyodunu değiştirir	Çizelge İşlemleri
ChartSymbol	Belirlenen çizelgenin sembol ismine dönüş yapar	Çizelge İşlemleri
ChartTimeOnDropped	Uzman Danışmanın veya betiğin iliştirildiği çizelge noktasının zaman koordinatına dönüş yapar	Çizelge İşlemleri
ChartTimePriceToXY	Çizelge koordinatlarını zaman/fiyat ifadesinden, X ve Y koordinatlarına dönüştürür	Çizelge İşlemleri
ChartWindowFind	Göstergelerin çizildiği alt pencerelerin sayısına dönüş yapar	Çizelge İşlemleri
ChartWindowOnDropped	Uzman Danışmanın veya betiğin iliştirildiği çizelge alt penceresinin indisine dönüş yapar	Çizelge İşlemleri
ChartXOnDropped	Uzman Danışmanın veya betiğin iliştirildiği çizelge noktasının X koordinatına dönüş yapar	Çizelge İşlemleri
ChartXYToTimePrice	Bir çizelgedeki X ve Y koordinatlarını, zaman ve fiyat değerlerine dönüştürür	Çizelge İşlemleri
ChartYOnDropped	Uzman Danışmanın veya betiğin iliştirildiği çizelge noktasının Y koordinatına dönüş yapar	Çizelge İşlemleri
CheckPointer	Nesne işaretçisinin tipine dönüş yapar	Yaygın Fonksiyonlar

Fonksiyon	Eylem	Bölüm
CLBufferCreate	Bir OpenCL tamponu oluşturur	OpenCL ile Çalışma
CLBufferFree	Bir OpenCL tamponunu siler	OpenCL ile Çalışma
CLBufferRead	OpenCL tamponunu bir diziye okur	OpenCL ile Çalışma
CLBufferWrite	Bir diziyi OpenCL tamponuna yazar	OpenCL ile Çalışma
CLContextCreate	Bir OpenCL bağlamı oluşturur	OpenCL ile Çalışma
CLContextFree	Bir OpenCL bağlamını siler	OpenCL ile Çalışma
CLExecute	Bir OpenCL programını çalıştırır	OpenCL ile Çalışma
CLGetDeviceInfo	OpenCL sürücüsünden aygıtın özelliğini alır	OpenCL ile Çalışma
CLGetInfoInteger	Bir OpenCL aygıtı veya nesnesi için bir tamsayı özelliğinin değerine dönüş yapar	OpenCL ile Çalışma
CLHandleType	Bir OpenCL tanıtıcı değerinin (handle) tipine, ENUM_OPENCL_HANDLE_TYPE sayımının değerlerinden biri şeklinde dönüş yapar	OpenCL ile Çalışma
CLKernelCreate	Bir OpenCL başlatma fonksiyonu oluşturur	OpenCL ile Çalışma
CLKernelFree	OpenCL başlatma fonksiyonunu siler	OpenCL ile Çalışma
CLProgramCreate	Bir kaynak kodundan OpenCL programı oluşturur	OpenCL ile Çalışma
CLProgramFree	Bir OpenCL programını siler	OpenCL ile Çalışma
CLSetKernelArg	OpenCL fonksiyonu için parametre ayarlar	OpenCL ile Çalışma
CLSetKernelArgMem	Bir OpenCL tamponunu OpenCL fonksiyonunun parametresi olarak ayarlar	OpenCL ile Çalışma
ColorToARGB	Bir rengin ARGB temsilinin alınması için, color tipini uint tipine dönüştürür.	Dönüşüm Fonksiyonları
ColorToString	Renk değerini, "R,G,B" şeklinde bir dizgiye dönüştürür	Dönüşüm Fonksiyonları
Comment	Çizelgenin sol üst köşesine bir yorumu çıktılar	Yaygın Fonksiyonlar

Fonksiyon	Eylem	Bölüm
CopyBuffer	Belirtilen bir göstergenin belirtilen bir tamponunu bir dizi içine kopyalar	Zaman Serilerine ve Göstergelere Erişim
CopyClose	Belirtilen sembol ve periyot için çubuk kapanış fiyatına dair geçmiş verisini bir diziye kopyalar	Zaman Serilerine ve Göstergelere Erişim
CopyHigh	Belirtilen sembol ve periyot için maksimal çubuk fiyatına dair geçmiş verisini bir diziye kopyalar	Zaman Serilerine ve Göstergelere Erişim
CopyLow	Belirtilen sembol ve periyot için minimal çubuk fiyatına dair geçmiş verisini bir diziye kopyalar	Zaman Serilerine ve Göstergelere Erişim
CopyOpen	Belirtilen sembol ve periyot için çubuk açılış fiyatına dair geçmiş verisini bir diziye kopyalar	Zaman Serilerine ve Göstergelere Erişim
CopyRates+	Belirtilen sembol ve periyot için Rates yapısının tarihsel verisini bir diziye kopyalar	Zaman Serilerine ve Göstergelere Erişim
CopyRealVolume	Belirtilen sembol ve periyot için alım-satım hacmine dair geçmiş verisini bir diziye kopyalar	Zaman Serilerine ve Göstergelere Erişim
CopySpread	Belirtilen sembol ve periyot için makas değerine dair geçmiş verisini bir diziye kopyalar	Zaman Serilerine ve Göstergelere Erişim
CopyTicks	Gets ticks accumulated by the terminal for the current working session into an array	Zaman Serilerine ve Göstergelere Erişim
CopyTickVolume	Belirtilen sembol ve periyot için tik hacmine dair geçmiş verisini bir diziye kopyalar	Zaman Serilerine ve Göstergelere Erişim
CopyTime	Belirtilen sembol ve periyot için çubuk açılış zamanına dair geçmiş verisini bir diziye kopyalar	Zaman Serilerine ve Göstergelere Erişim
cos	Bir sayının kosinüsüne dönüş yapar	Matematik Fonksiyonları
CryptDecode	Performs the inverse transformation of the data from array	Yaygın Fonksiyonlar
CryptEncode	Transforms the data from array with the specified method	Yaygın Fonksiyonlar

Fonksiyon	Eylem	Bölüm
CustomSymbolCreate	Belirtilen grubun içinde belirtilen isimde bir sembol oluşturur	Kullanıcı-tanımlı semboller
CustomSymbolDelete	Belirtilen isme sahip olan kullanıcı-tanımlı sembolü siler	Kullanıcı-tanımlı semboller
CustomSymbolSetInteger	Kullanıcı-tanımlı sembolün tamsayı tipli bir özelliğini ayarlar	Kullanıcı-tanımlı semboller
CustomSymbolSetDouble	Kullanıcı-tanımlı sembolün reel tipli bir özelliğini ayarlar	Kullanıcı-tanımlı semboller
CustomSymbolSetString	Kullanıcı-tanımlı sembolün dizgi tipli bir özelliğini ayarlar	Kullanıcı-tanımlı semboller
CustomSymbolSetMarginRate	Kullanıcı-tanımlı bir sembol için emir türüne ve yönüne göre ilgili teminat oranlarını ayarlar	Kullanıcı-tanımlı semboller
CustomSymbolSetSessionQuote	Belirtilen sembol ve gün değeri için fiyatlandırma seansının başlangıç ve bitiş zamanlarını ayarlar	Kullanıcı-tanımlı semboller
CustomSymbolSetSessionTrade	Belirtilen sembol ve gün değeri için işlem seansının başlangıç ve bitiş zamanlarını ayarlar	Kullanıcı-tanımlı semboller
CustomRatesDelete	Kullanıcı-tanımlı sembolün fiyat geçmişi üzeride belirtilen zaman aralığındaki tüm çubuk verilerini siler.	Kullanıcı-tanımlı semboller
CustomRatesReplace	Kullanıcı-tanımlı sembolün veri geçmişinde, belirtilen zaman aralığındaki verileri 'MqlRates' tipli dizinin verileri ile değiştirir	Kullanıcı-tanımlı semboller
CustomRatesUpdate	Kullanıcı-tanımlı bir sembolün veri geçmişindeki eksik verileri ekler ve mevcut verileri 'MqlRates' tipli diziden kopyalayarak değiştirir	Kullanıcı-tanımlı semboller
CustomTicksAdd	Kullanıcı-tanımlı bir sembol için MqlTick tipli bir diziden veri ekler. Kullanıcı-tanımlı sembol Piyasa Gözlemi penceresinde seçilmiş olmalıdır	Kullanıcı-tanımlı semboller
CustomTicksDelete	Kullanıcı-tanımlı sembolün belirtilen aralıktaki geçmiş tik verilerini siler	Kullanıcı-tanımlı semboller

Fonksiyon	Eylem	Bölüm
CustomTicksReplace	Kullanıcı-tanımlı sembolün belirtilen zaman aralığındaki geçmiş verilerini 'MqlTick' tipli diziden alınan verilerle değiştirir	Kullanıcı-tanımlı semboller
CustomBookAdd	Kullanıcı-tanımlı sembol için Piyasa Derinliği durumunu aktarır	Kullanıcı-tanımlı semboller
DatabaseOpen	Belirtilen dosyada veritabanı açar veya oluşturur	Veritabanlarıyla çalışma
DatabaseClose	Veritabanını kapatır	Veritabanlarıyla çalışma
DatabaseImport	Verileri bir dosyadan tabloya aktarır	Veritabanlarıyla çalışma
DatabaseExport	Bir tablo veya SQL isteği yürütme sonucunu bir CSV dosyasına aktarır	Veritabanlarıyla çalışma
DatabasePrint	Uzman Danışmanlar günlüğüne bir tablo veya SQL isteği yürütme sonucu yazdırır	Veritabanlarıyla çalışma
DatabaseTableExists	Veritabanında tablonun varlığını kontrol eder	Veritabanlarıyla çalışma
DatabaseExecute	Belirtilen veritabanına istek yürütür	Veritabanlarıyla çalışma
DatabasePrepare	Daha sonra DatabaseRead() kullanılarak yürütülebilen bir istek tanıtıcı değeri oluşturur	Veritabanlarıyla çalışma
DatabaseReset	DatabasePrepare() çağrıldıktan sonra olduğu gibi isteği sıfırlar	Veritabanlarıyla çalışma
DatabaseBind	İstekte bir parametre değeri ayarlar	Veritabanlarıyla çalışma
DatabaseBindArray	Diziyi parametre değeri olarak ayarlar	Veritabanlarıyla çalışma
DatabaseRead	İstek sonucunda bir sonraki girdiye gider	Veritabanlarıyla çalışma
DatabaseFinalize	DatabasePrepare()'da oluşturulan isteği kaldırır	Veritabanlarıyla çalışma
DatabaseTransactionBegin	İşlem yürütmeyi başlatır	Veritabanlarıyla çalışma
DatabaseTransactionCommit	İşlem yürütmeyi tamamlar	Veritabanlarıyla çalışma
DatabaseTransactionRollback	İşlemleri geri alır	Veritabanlarıyla çalışma
DatabaseColumnsCount	İstekteki alan sayısını elde eder	Veritabanlarıyla çalışma

Fonksiyon	Eylem	Bölüm
DatabaseColumnName	Alan adını indekse göre elde eder	Veritabanlarıyla çalışma
DatabaseColumnType	Alan tipini indekse göre elde eder	Veritabanlarıyla çalışma
DatabaseColumnSize	Alan boyutunu bayt cinsinden elde eder	Veritabanlarıyla çalışma
DatabaseColumnText	Alan değerini geçerli kayıttan dizge olarak elde eder	Veritabanlarıyla çalışma
DatabaseColumnInteger	Geçerli kayıttan int tipi değeri elde eder	Veritabanlarıyla çalışma
DatabaseColumnLong	Geçerli kayıttan long tipi değeri elde eder	Veritabanlarıyla çalışma
DatabaseColumnDouble	Geçerli kayıttan double tipi değeri elde eder	Veritabanlarıyla çalışma
DatabaseColumnBlob	Alan değerini geçerli kayıttan dizi olarak elde eder	Veritabanlarıyla çalışma
DebugBreak	Hata ayıklamadaki program kırılma noktası	Yaygın Fonksiyonlar
Digits	Mevcut çizelge sembolünün fiyat değeri çözünürlüğünü belirleyen ondalık basamak sayısına dönüş yapar	Durun Kontrolü
DoubleToString	Sayısal bir değeri, belirtilen çözünürlükle bir metne dönüştürür	Dönüşüm Fonksiyonları
DXContextCreate	Belirli bir boyuttaki kareleri render almak için bir grafik içeriği oluşturur	DirectX ile çalışma
DXContextSetSize	DXContextCreate()'de oluşturulan grafik içeriğinin kare boyutunu değiştirir	DirectX ile çalışma
DXContextSetSize	DXContextCreate()'de oluşturulan grafik içeriğinin kare boyutunu elde eder	DirectX ile çalışma
DXContextClearColors	Render alma tamponu için tüm pikselleri belirtilen renge ayarlar	DirectX ile çalışma
DXContextClearDepth	Derinlik tamponunu temizler	DirectX ile çalışma
DXContextGetColors	Grafik içeriğinden belirtilen boyutta ve ofsette bir görüntü elde eder	DirectX ile çalışma

Fonksiyon	Eylem	Bölüm
DXContextGetDepth	Oluşturulan bir karenin derinlik tamponunu elde eder	DirectX ile çalışma
DXBufferCreate	Veri dizisine dayalı olarak belirtilen tipte bir tampon oluşturur	DirectX ile çalışma
DXTextureCreate	Aktarılan görüntüden kesilmiş belirli bir boyuttaki dikdörtgenden 2D bir doku oluşturur	DirectX ile çalışma
DXInputCreate	Gölgelendirici girdilerini oluşturur	DirectX ile çalışma
DXInputSet	Gölgelendirici girdilerini ayarlar	DirectX ile çalışma
DXShaderCreate	Belirtilen tipte bir gölgelendirici oluşturur	DirectX ile çalışma
DXShaderSetLayout	Köşe gölgelendirici için köşe düzenini ayarlar	DirectX ile çalışma
DXShaderInputsSet	Gölgelendirici girdilerini ayarlar	DirectX ile çalışma
DXShaderTexturesSet	Gölgelendirici dokularını ayarlar	DirectX ile çalışma
DXDraw	DXBufferSet()'te ayarlanan köşe tamponunun köşe noktalarını render alır	DirectX ile çalışma
DXDrawIndexed	DXBufferSet() indeks tamponu tarafından tanımlanan grafik ilkellerini oluşturur	DirectX ile çalışma
DXPrimiveTopologySet	DXDrawIndexed()'i kullanarak render almak için ilkellerin tipini ayarlar	DirectX ile çalışma
DXBufferSet	Geçerli render işlemi için bir tampon ayarlar	DirectX ile çalışma
DXShaderSet	Render alma için gölgelendirici ayarlar	DirectX ile çalışma
DXHandleType	Tanıtıcı değer tipini geri döndürür	DirectX ile çalışma
DXRelease	Tanıtıcı değeri serbest bırakır	DirectX ile çalışma
EnumToString	Herhangi tipteki bir sayım değerini dizgiye dönüştürür	Dönüşüm Fonksiyonları
EventChartCustom	Belirtilen çizelge için özel bir olay oluşturur	Olaylarla Çalışma
EventKillTimer	Mevcut çizelgede, zamanlayıcı tarafından olayların oluşturulmasını durdurur	Olaylarla Çalışma

Fonksiyon	Eylem	Bölüm
EventSetMillisecondTimer	Yüksek çözünürlüklü zamanlayıcının olay oluşturucusunu, 1 saniyeden az olan bir periyot ile mevcut çizelge için çalıştırır	Olaylarla Çalışma
EventSetTimer	Zamanlayıcı olayının oluşturucusunu, belirtilen periyotta mevcut çizelge için başlatır	Olaylarla Çalışma
exp	Bir sayının eksponentine dönüş yapar	Matematik Fonksiyonları
ExpertRemove	Uzman Danışmanı durdurur ve çizelgeden kaldırır	Yaygın Fonksiyonlar
fabs	Belirtilen sayısal değerin mutlak değerine dönüş yapar	Matematik Fonksiyonları
FileClose	Açılmış bir dosyayı kapatır	Dosya Fonksiyonları
FileCopy	Orjinal dosyayı, yerel veya paylaşımlı bir klasörden, başka bir dosyaya kopyalar	Dosya Fonksiyonları
FileDelete	Belirtilen bir dosyayı siler	Dosya Fonksiyonları
FileFindClose	Arama işleyicisini kapatır	Dosya Fonksiyonları
FileFindFirst	Belirtilen filtreye göre belli bir konumda dosya araması başlatır	Dosya Fonksiyonları
FileFindNext	FileFindFirst() fonksiyonu ile yapılan aramayı devam ettirir	Dosya Fonksiyonları
FileFlush	Giriş/çıkış tamponunda kalan tüm veriyi bir diske yazar	Dosya Fonksiyonları
FileGetInteger	Dosyaya dair bir tamsayı özelliğini alır	Dosya Fonksiyonları
FileIsEnding	Okuma sürecinde bir dosyanın sonunu tanımlar	Dosya Fonksiyonları
FileIsExist	Bir dosyanın varlığını kontrol eder	Dosya Fonksiyonları
FileIsLineEnding	Okuma sürecinde bir metin dosyasındaki bir satırın sonunu tanımlar	Dosya Fonksiyonları
FileMove	Bir dosyayı taşır veya yeniden adlandırır	Dosya Fonksiyonları

Fonksiyon	Eylem	Bölüm
FileOpen	Belirli bir isme ve bayrağa sahip olan dosyayı açar	Dosya Fonksiyonları
FileReadArray	BIN tipi bir dosyadan, string harici herhangi tipteki dizileri okur	Dosya Fonksiyonları
FileReadBool	CSV dosyasındaki bir dizgiyi ayırıcı işarete kadar (veya satır sonuna kadar) okur ardından okunan dizgiyi bool tipine dönüştürür	Dosya Fonksiyonları
FileReadDatetime	CSV dosyasındaki, "YYYY.MM.DD HH:MM:SS", "YYYY.MM.DD" veya "HH:MM:SS" biçimlerinden birindeki dizgiyi okur ve bir datetime değerine dönüştürür	Dosya Fonksiyonları
FileReadDouble	Dosya işaretçisinin mevcut konumundan double tipi değeri okur	Dosya Fonksiyonları
FileReadFloat	Dosya işaretçisinin mevcut konumundan float tipi bir değeri okur	Dosya Fonksiyonları
FileReadInteger	Dosya işaretçisinin mevcut konumundan short veya char tipi bir değeri okur	Dosya Fonksiyonları
FileReadLong	Dosya işaretçisinin mevcut konumundan long tipi bir değeri okur	Dosya Fonksiyonları
FileReadNumber	CSV dosyasındaki bir dizgiyi, mevcut konumdan ayırıcı işarete kadar (veya satır sonuna kadar) okur, ardından bu dizgiyi double tipine dönüştürür	Dosya Fonksiyonları
FileReadString	Bir dosyadaki dizgiyi, dosya işaretçisinin mevcut konumundan başlayarak okur	Dosya Fonksiyonları
FileReadStruct	Bir yapıya parametre olarak geçirilmiş ikili dosyanın içeriğini okur	Dosya Fonksiyonları
FileSeek	Dosya işaretçisinin konumunu, belirtilen bayt sayısı ile, belirlenen konuma taşır	Dosya Fonksiyonları

Fonksiyon	Eylem	Bölüm
FileSize	Söz konusu açık dosyanın boyutuna dönüş yapar	Dosya Fonksiyonları
FileTell	Söz konusu açık dosyanın, mevcut dosya işaretçisi konumuna dönüş yapar	Dosya Fonksiyonları
FileWrite	CSV veya TXT tipi dosyaya veri yazar	Dosya Fonksiyonları
FileWriteArray	BIN tipi bir dosyaya herhangi bir tipte (string hariç) diziler yazar	Dosya Fonksiyonları
FileWriteDouble	İkili bir dosya içine, işaretçinin mevcut konumundan başlayarak double tipi bir değer yazar	Dosya Fonksiyonları
FileWriteFloat	İkili bir dosya içine, işaretçinin mevcut konumundan başlayarak float tipi bir değer yazar file	Dosya Fonksiyonları
FileWriteInteger	İkili bir dosya içine, işaretçinin mevcut konumundan başlayarak int tipi bir değer yazar file	Dosya Fonksiyonları
FileWriteLong	İkili bir dosya içine, işaretçinin mevcut konumundan başlayarak long tipi bir değer yazar	Dosya Fonksiyonları
FileWriteString	Bir BIN veya TXT dosyasının içine, işaretçinin mevcut konumundan başlayarak string tipi bir değer yazar	Dosya Fonksiyonları
FileWriteStruct	İkili bir dosya içine, işaretçinin mevcut konumundan başlayarak bir yapının içeriklerini yazar	Dosya Fonksiyonları
floor	Nümerik bir değer için en yakın küçük tamsayı değerine dönüş yapar	Matematik Fonksiyonları
fmax	İki sayısal sayı arasındaki en büyük değere dönüş yapar	Matematik Fonksiyonları
fmin	İki sayısal sayı arasındaki en küçük değere dönüş yapar	Matematik Fonksiyonları
fmod	İki sayının bölümünden kalan reel sayıya dönüş yapar	Matematik Fonksiyonları
FolderClean	Belirtilen klasördeki tüm dosyaları siler	Dosya Fonksiyonları

Fonksiyon	Eylem	Bölüm
FolderCreate	Files dizini içinde (common_flag değerine bağlı olarak) bir klasör oluşturur	Dosya Fonksiyonları
FolderDelete	Seçilen klasörü siler. Klasör boş değilse silinemez	Dosya Fonksiyonları
FrameAdd	Verilerle birlikte bir çerçeve ekler	Optimizasyon Sonuçları ile Çalışma
FrameFilter	Çerçeve okuma filtresini ayarlar ve işaretçiyi başlangıca taşır	Optimizasyon Sonuçları ile Çalışma
FrameFirst	Çerçevenin okuma işaretçisini başlangıca taşır ve önceden ayarlanmış olan filtreyi siler	Optimizasyon Sonuçları ile Çalışma
FrameInputs	Çerçevenin şekillendirildiği giriş parametresini alır	Optimizasyon Sonuçları ile Çalışma
FrameNext	Bir çerçeveyi okur ve işaretçiyi bir sonrakine taşır	Optimizasyon Sonuçları ile Çalışma
GetLastError	Son hataya dönüş yapar	Durun Kontrolü
GetPointer	Nesne işaretçisine dönüş yapar	Yaygın Fonksiyonlar
GetTickCount	Sistemin başlatılmasından bu yana geçen milisaniyelerin sayısına dönüş yapar	Yaygın Fonksiyonlar
GlobalVariableCheck	Belirtilen isimdeki bir global değişkenin varlığını kontrol eder	Terminalin Global Değişkenleri
GlobalVariableDel	Bir global değişkeni siler	Terminalin Global Değişkenleri
GlobalVariableGet	Bir global değişkenin değerine dönüş yapar	Terminalin Global Değişkenleri
GlobalVariableName	Global değişkenler listesindeki sıra sayısına göre global bir değişken ismine dönüş yapar	Terminalin Global Değişkenleri
GlobalVariablesDeleteAll	Belirtilen ön eke sahip global değişkeni siler	Terminalin Global Değişkenleri
GlobalVariableSet	Bir global değişkene, yeni bir değer ayarlar	Terminalin Global Değişkenleri
GlobalVariableSetOnCondition	Bir global değişkene, koşula bağlı olarak yeni bir değer ayarlar	Terminalin Global Değişkenleri
GlobalVariablesFlush	Tüm global değişkenlerin değerlerini zorla diske yazar	Terminalin Global Değişkenleri

Fonksiyon	Eylem	Bölüm
GlobalVariablesTotal	Global değişkenlerin toplam sayısına dönüş yapar	Terminalin Global Değişkenleri
GlobalVariableTemp	Bir global değişkene, terminalin sadece mevcut oturumunda geçerli olacak yeni bir değer ayarlar	Terminalin Global Değişkenleri
GlobalVariableTime	Global değişkene yapılan son erişimin zamanına dönüş yapar	Terminalin Global Değişkenleri
HistoryDealGetDouble	Geçmişteki bir işlemin istenen (double) özelliğine dönüş yapar	Alım-Satım Fonksiyonları
HistoryDealGetInteger	Geçmişteki bir işlemin istenen (datetime veya int) özelliğine dönüş yapar	Alım-Satım Fonksiyonları
HistoryDealGetString	Geçmişteki bir işlemin istenen (string) özelliğine dönüş yapar	Alım-Satım Fonksiyonları
HistoryDealGetTicket	Geçmişteki bir işlemin fişine dönüş yapar	Alım-Satım Fonksiyonları
HistoryDealSelect	Uygun fonksiyonlarla yapılacak daha sonraki çağrılar için geçmişteki bir işlemi seçer	Alım-Satım Fonksiyonları
HistoryDealsTotal	Geçmişteki işlemlerin sayısına dönüş yapar	Alım-Satım Fonksiyonları
HistoryOrderGetDouble	Geçmişteki bir emrin istenen (double) özelliğine dönüş yapar	Alım-Satım Fonksiyonları
HistoryOrderGetInteger	Geçmişteki bir emrin istenen (datetime veya int) özelliğine dönüş yapar	Alım-Satım Fonksiyonları
HistoryOrderGetString	Geçmişteki bir emrin istenen (string) özelliğine dönüş yapar	Alım-Satım Fonksiyonları
HistoryOrderGetTicket	Geçmişteki karşılık gelen emrin fişine dönüş yapar	Alım-Satım Fonksiyonları
HistoryOrderSelect	Daha sonraki işler için geçmişteki bir emri seçer	Alım-Satım Fonksiyonları
HistoryOrdersTotal	Geçmişteki emirlerin sayısına dönüş yapar	Alım-Satım Fonksiyonları
HistorySelect	Sunucu zamanının belirli bir periyodu için, emir ve faaliyet geçmişini düzeltir	Alım-Satım Fonksiyonları

Fonksiyon	Eylem	Bölüm
iBars	Tarihte mevcut karşılık gelen sembol ve dönemin çubuk sayısını döndürür	Zaman Serilerine ve Göstergelere Erişim
iBarShift	Zamana göre bar arama. İşlev, belirtilen zamana karşılık gelen çubuğun dizinini döndürür	Zaman Serilerine ve Göstergelere Erişim
iClose	İlgili grafikteki çubuğun Kapanış fiyatını ('shift' parametresiyle gösterilir) döndürür	Zaman Serilerine ve Göstergelere Erişim
iHigh	İlgili grafikteki çubuğun Yüksek fiyatını ('shift' parametresiyle gösterilir) döndürür	Zaman Serilerine ve Göstergelere Erişim
iHighest	İlgili grafikte bulunan en yüksek değerin indeksini döndürür (geçerli çubuğa göre)	Zaman Serilerine ve Göstergelere Erişim
iLow	İlgili grafikteki çubuğun Düşük fiyatını ('shift' parametresiyle gösterilir) döndürür	Zaman Serilerine ve Göstergelere Erişim
iLowest	İlgili grafikte bulunan en küçük değerin indeksini döndürür (geçerli çubuğa göre)	Zaman Serilerine ve Göstergelere Erişim
iOpen	İlgili grafikteki çubuğun Açılış fiyatını ('shift' parametresiyle gösterilir) döndürür	Zaman Serilerine ve Göstergelere Erişim
iTime	İlgili grafikteki çubuğun açılış zamanını ('shift' parametresiyle gösterilir) döndürür	Zaman Serilerine ve Göstergelere Erişim
iTickVolume	İlgili grafikteki çubuğun tik hacmini ('shift' parametresi ile gösterilir) döndürür	Zaman Serilerine ve Göstergelere Erişim
iRealVolume	İlgili grafikteki çubuğun gerçek hacmini ('shift' parametresiyle gösterilir) döndürür	Zaman Serilerine ve Göstergelere Erişim
iVolume	İlgili grafikteki çubuğun tik hacmini ('shift' parametresi ile gösterilir) döndürür	Zaman Serilerine ve Göstergelere Erişim
iSpread	İlgili grafikteki çubuğun spread değerini ('shift' parametresiyle gösterilir) döndürür	Zaman Serilerine ve Göstergelere Erişim
iAC	Accelerator Oscillator	Teknik Göstergeler

Fonksiyon	Eylem	Bölüm
iAD	Accumulation/Distribution	Teknik Göstergeler
iADX	Average Directional Index	Teknik Göstergeler
iADXWilder	Average Directional Index by Welles Wilder	Teknik Göstergeler
iAlligator	Alligator	Teknik Göstergeler
iAMA	Adaptive Moving Average	Teknik Göstergeler
iAO	Awesome Oscillator	Teknik Göstergeler
iATR	Average True Range	Teknik Göstergeler
iBands	Bollinger Bands®	Teknik Göstergeler
iBearsPower	Bears Power	Teknik Göstergeler
iBullsPower	Bulls Power	Teknik Göstergeler
iBWMFI	Market Facilitation Index by Bill Williams	Teknik Göstergeler
iCCI	Commodity Channel Index	Teknik Göstergeler
iChaikin	Chaikin Oscillator	Teknik Göstergeler
iCustom	Özel gösterge	Teknik Göstergeler
iDEMA	Double Exponential Moving Average	Teknik Göstergeler
iDeMarker	DeMarker	Teknik Göstergeler
iEnvelopes	Envelopes	Teknik Göstergeler
iForce	Force Index	Teknik Göstergeler
iFractals	Fractals	Teknik Göstergeler
iFrAMA	Fractal Adaptive Moving Average	Teknik Göstergeler
iGator	Gator Oscillator	Teknik Göstergeler
iIchimoku	Ichimoku Kinko Hyo	Teknik Göstergeler
iMA	Moving Average	Teknik Göstergeler
iMACD	Moving Averages Convergence-Divergence	Teknik Göstergeler
iMFI	Money Flow Index	Teknik Göstergeler
iMomentum	Momentum	Teknik Göstergeler
IndicatorCreate	MqlParam tipi parametrelerden oluşmuş bir dizi ile oluşturulan	Zaman Serilerine ve Göstergelere Erişim

Fonksiyon	Eylem	Bölüm
	belirli bir teknik göstergenin tanıtıcı değerine dönüş yapar	
IndicatorParameters	Belirtilen tanıtıcı değere bağlı olarak, gösterge giriş parametrelerinin sayısına, değerlerine ve tiplerine dönüş yapar	Zaman Serilerine ve Göstergelere Erişim
IndicatorRelease	Gösterge tanıtıcı değerini kaldırır ve başka biri tarafından kullanılmıyorsa hesaplama bloğunu serbest bırakır	Zaman Serilerine ve Göstergelere Erişim
IndicatorSetDouble	double tipi gösterge özelliğinin değerini ayarlar	Özel Göstergeler
IndicatorSetInteger	int tipi gösterge özelliğinin değerini ayarlar	Özel Göstergeler
IndicatorSetString	string tipi gösterge özelliğinin değerini ayarlar	Özel Göstergeler
IntegerToString	int tipi bir değeri önceden ayarlanmış uzunlukta bir dizgiye dönüştürür	Dönüşüm Fonksiyonları
iOBV	On Balance Volume	Teknik Göstergeler
iOsMA	Moving Average of Oscillator (MACD histogram)	Teknik Göstergeler
iRSI	Relative Strength Index	Teknik Göstergeler
iRVI	Relative Vigor Index	Teknik Göstergeler
iSAR	Parabolic Stop And Reverse System	Teknik Göstergeler
IsStopped	Bir MQL5 programı işlemi durdurma emri almışsa, 'true' değerine dönüş yapar	Durun Kontrolü
iStdDev	Standard Deviation	Teknik Göstergeler
iStochastic	Stochastic Oscillator	Teknik Göstergeler
iTEMA	Triple Exponential Moving Average	Teknik Göstergeler
iTriX	Triple Exponential Moving Averages Oscillator	Teknik Göstergeler
iVIDyA	Variable Index Dynamic Average	Teknik Göstergeler
iVolumes	Volumes	Teknik Göstergeler

Fonksiyon	Eylem	Bölüm
iWPR	Williams' Percent Range	Teknik Göstergeler
log	Sayının normal logaritmasına dönüş yapar	Matematik Fonksiyonları
log10	Bir sayının 10 tabanındaki logaritmasına dönüş yapar	Matematik Fonksiyonları
MarketBookAdd	Seçilen bir sembol için Piyasa Derinliğinin açılmasını ve değişimler ile ilgili bilgi alınmasını sağlar	Piyasa Bilgisi
MarketBookGet	Belirtilen bir sembolün Piyasa Derinliği kayıtlarını içeren MqlBookInfo yapı dizisine dönüş yapar	Piyasa Bilgisi
MarketBookRelease	Seçilen bir sembol için Piyasa Derinliğinin kapanmasını sağlayıp, değişimler ile ilgili bilgi alınmasını sonlandırır	Piyasa Bilgisi
MathAbs	Belirtilen sayısal değerin mutlak değerine dönüş yapar	Matematik Fonksiyonları
MathArccos	Bir x değişkeninin ark kosinüs değerine, radyan cinsinden dönüş yapar	Matematik Fonksiyonları
MathArcsin	Bir x değişkeninin ark sinüs değerine, radyan cinsinden dönüş yapar	Matematik Fonksiyonları
MathArctan	Bir x değişkeninin ark tanjant değerine, radyan cinsinden dönüş yapar	Matematik Fonksiyonları
MathCeil	Nümerik bir değer için en yakın büyük tamsayı değerine dönüş yapar	Matematik Fonksiyonları
MathCos	Bir sayının kosinüsüne dönüş yapar	Matematik Fonksiyonları
MathExp	Bir sayının eksponentine dönüş yapar	Matematik Fonksiyonları
MathFloor	Nümerik bir değer için en yakın küçük tamsayı değerine dönüş yapar	Matematik Fonksiyonları
MathIsValidNumber	Bir reel sayının doğruluğunu kontrol eder	Matematik Fonksiyonları

Fonksiyon	Eylem	Bölüm
MathLog	Sayının normal logaritmasına dönüş yapar	Matematik Fonksiyonları
MathLog10	Bir sayının 10 tabanındaki logaritmasına dönüş yapar	Matematik Fonksiyonları
MathMax	İki sayısal sayı arasındaki en büyük değere dönüş yapar	Matematik Fonksiyonları
MathMin	İki sayısal sayı arasındaki en küçük değere dönüş yapar	Matematik Fonksiyonları
MathMod	İki sayının bölümünden kalan reel sayıya dönüş yapar	Matematik Fonksiyonları
MathPow	Bir sayının kuvvetini alır	Matematik Fonksiyonları
MathRand	0 ile 32767 arasında bir (psuedo) rassal sayıya dönüş yapar	Matematik Fonksiyonları
MathRound	Değeri, en yakın tamsayıya yuvarlar	Matematik Fonksiyonları
MathSin	Bir sayının sinüsüne dönüş yapar	Matematik Fonksiyonları
MathSqrt	Sayının kareköküne dönüş yapar	Matematik Fonksiyonları
MathSrand	Bir dizi (psuedo) rassal sayı oluşturmak için başlangıç değeri ayarlar	Matematik Fonksiyonları
MathTan	Bir sayının tanjantını alır	Matematik Fonksiyonları
MessageBox	Bir mesaj kutusu oluşturur, görüntüler ve yönetir	Yaygın Fonksiyonlar
MQLInfoInteger	Çalışmakta olan bir MQL5 programının karşılık gelen özelliğinin tamsayı değerine dönüş yapar	Durun Kontrolü
MQLInfoString	Çalışmakta olan bir MQL5 programının karşılık gelen özelliğinin string tipli değerine dönüş yapar	Durun Kontrolü
MT5Initialize	MetaTrader 5 terminali ile bağlantı kur	Python için MetaTrader
MT5Shutdown	MetaTrader 5 terminaline önceden kurulmuş olan bağlantıyı kapat.	Python için MetaTrader
MT5TerminalInfo	Bağlantı kurulmuş olan MetaTrader 5 terminalinin	Python için MetaTrader

Fonksiyon	Eylem	Bölüm
	durumunu ve parametrelerini elde et	
MT5Version	MetaTrader 5 terminal versiyonunu geri döndür	Python için MetaTrader
MT5CopyRatesFrom	Belirtilen tarihten itibaren MetaTrader 5 terminalinden barlar elde et	Python için MetaTrader
MT5CopyRatesFromPos	Belirtilen dizinden itibaren MetaTrader 5 terminalinden barlar elde et	Python için MetaTrader
MT5CopyRatesRange	MetaTrader 5 terminalinden belirtilen tarih aralığındaki barları elde et	Python için MetaTrader
MT5CopyTicksFrom	Belirtilen tarihten itibaren MetaTrader 5 terminalinden tikler elde et	Python için MetaTrader
MT5CopyTicksRange	MetaTrader 5 terminalinden belirtilen tarih aralığındaki tikleri elde et	Python için MetaTrader
NormalizeDouble	Bir kayan noktalı sayıyı, belirlenen çözünürlük ile yuvarlar	Dönüşüm Fonksiyonları
ObjectCreate	Belirtilen çizelge üzerinde belirlenen tipte bir nesne oluşturur	Nesne Fonksiyonları
ObjectDelete	Belirtilen çizelge (veya çizelge alt-penceresi) üzerinde belirtilen tipte bir nesneyi siler	Nesne Fonksiyonları
ObjectFind	Belirtilen tanımlayıcı ve ismi kullanarak bir nesneyi arar	Nesne Fonksiyonları
ObjectGetDouble	Karşılık gelen nesne özelliğinin double tipi değerine dönüş yapar	Nesne Fonksiyonları
ObjectGetInteger	Karşılık gelen nesne özelliğinin tamsayı değerine dönüş yapar	Nesne Fonksiyonları
ObjectGetString	Karşılık gelen nesne özelliğinin string tipli değerine dönüş yapar	Nesne Fonksiyonları
ObjectGetTimeByValue	Nesnenin belirtilen fiyat değeri için zaman değerine dönüş yapar	Nesne Fonksiyonları
ObjectGetValueByTime	Nesnenin belirtilen zaman değeri için fiyat değerine dönüş yapar	Nesne Fonksiyonları

Fonksiyon	Eylem	Bölüm
ObjectMove	Bir nesnenin tutturma noktası koordinatlarını değiştirir	Nesne Fonksiyonları
ObjectName	Belirtilen çizelge (veya çizelge alt-penceresi) üzerinde belirtilen tipte bir nesnenin ismine dönüş yapar	Nesne Fonksiyonları
ObjectsDeleteAll	Belirtilen çizelge (veya çizelge alt-penceresi) üzerinde belirtilen tipteki tüm nesneleri siler	Nesne Fonksiyonları
ObjectSetDouble	Karşılık gelen nesne özelliğinin değerini ayarlar	Nesne Fonksiyonları
ObjectSetInteger	Karşılık gelen nesne özelliğinin değerini ayarlar	Nesne Fonksiyonları
ObjectSetString	Karşılık gelen nesne özelliğinin değerini ayarlar	Nesne Fonksiyonları
ObjectsTotal	Belirtilen çizelge (veya çizelge alt-penceresi) üzerinde belirtilen tipteki nesnelerin sayısına dönüş yapar	Nesne Fonksiyonları
OnStart	Fonksiyon, Start olayının skriptte ayarlanan bazı eylemleri gerçekleştirmek için oluştuğunda çağrılır.	Olay yönetimi
OnInit	Fonksiyon, çalışan bir MQL5 programını başlatmak için Init olayı oluştuğunda göstergeler ve uzman danışmanlarda çağrılır.	Olay yönetimi
OnDeinit	Fonksiyon, çalışan bir MQL5 programını sonlandırmak için Deinit olayı oluştuğunda göstergeler ve uzman danışmanlarda çağrılır.	Olay yönetimi
OnTick	Fonksiyon, yeni bir fiyatı(tik) işlemek için NewTick olayı oluştuğunda uzman danışmanlarda çağrılır.	Olay yönetimi
OnCalculate	Fonksiyon, fiyat bilgisinin değişimini işlemek için Calculate olayı oluştuğunda göstergelerde çağrılır.	Olay yönetimi

Fonksiyon	Eylem	Bölüm
OnTimer	Fonksiyon, Timer periyodik olayının terminal tarafından sabit zaman aralıklarında oluşturulmasıyla göstergeler ve uzman danışmanlarda çağrılır.	Olay yönetimi
OnTrade	Fonksiyon, bir işlem sunucusu üzerinde bir alım-satım işleminin sonunda oluşturulan Trade olayı sırasında uzman danışmanlarda çağrılır.	Olay yönetimi
OnTradeTransaction	Fonksiyon, bir işlem isteği gerçekleştirim sonuçlarını işlemek için TradeTransaction olayı oluştuğunda uzman danışmanlarda çağrılır.	Olay yönetimi
OnBookEvent	Fonksiyon, piyasa derinliğindeki değişiklikleri işlemek için BookEvent olayı oluştuğunda uzman danışmanlarda çağrılır.	Olay yönetimi
OnChartEvent	Fonksiyon, bir MQL5 programı veya bir kullanıcı tarafından yapılan grafik değişikliklerini işlemek için ChartEvent olayı oluştuğunda göstergelerde ve uzman danışmanlarda çağrılır.	Olay yönetimi
OnTester	Fonksiyon, bir uzman danışmanın geçmiş veriler üzerinde test edilmesinden sonra gerekli eylemleri gerçekleştirmek için Tester olayı oluştuğunda uzman danışmanlarda çağrılır.	Olay yönetimi
OnTesterInit	Fonksiyon, strateji sınayıcısında optimizasyondan önce gerekli eylemleri gerçekleştirmek için TesterInit olayı oluştuğunda uzman danışmanlarda çağrılır.	Olay yönetimi
OnTesterDeinit	Fonksiyon, strateji sınayıcısında uzman danışman optimizasyonundan sonra TesterDeinit olayı oluştuğunda uzman danışmanlarda çağrılır.	Olay yönetimi
OnTesterPass	Fonksiyon, strateji sınayıcısında uzman danışman optimizasyonu sırasında yeni bir veri	Olay yönetimi

Fonksiyon	Eylem	Bölüm
	çerçevesinin gelişini işlemek için TesterPass olayı oluştuğunda uzman danışmanlarda çağrılır.	
OrderCalcMargin	Belirtilen emir tipi için, teminat para birimi cinsinden gereken teminat miktarını hesaplar	Alım-Satım Fonksiyonları
OrderCalcProfit	Geçirilen parametrelere göre teminat para birimi cinsinden kar hesaplar	Alım-Satım Fonksiyonları
OrderCheck	İstenen alım-satım işlemini gerçekleştirmek için yeterli fon olup olmadığını kontrol eder.	Alım-Satım Fonksiyonları
OrderGetDouble	Bir emrin istenen (double) özelliğine dönüş yapar	Alım-Satım Fonksiyonları
OrderGetInteger	Bir emrin istenen (datetime veya int) özelliğine dönüş yapar	Alım-Satım Fonksiyonları
OrderGetString	Bir emrin istenen (string) özelliğine dönüş yapar	Alım-Satım Fonksiyonları
OrderGetTicket	Söz konusu emrin fişine dönüş yapar	Alım-Satım Fonksiyonları
OrderSelect	Daha sonraki işler için bir emri seçer	Alım-Satım Fonksiyonları
OrderSend	Sunucuya alım-satım istekleri gönderir	Alım-Satım Fonksiyonları
OrderSendAsync	Asenkron biçimde, sunucu cevabını beklemeden alım-satım istekleri gönderir	Alım-Satım Fonksiyonları
OrdersTotal	Emir sayısına dönüş yapar	Alım-Satım Fonksiyonları
ParameterGetRange	Bir Uzman Danışmanın sınayıcıdaki optimizasyonu sırasında girdi değişkeni için, veri aralığı ve değişim birimi hakkında bilgi alır	Optimizasyon Sonuçları ile Çalışma
ParameterSetRange	Bir Uzman Danışmanın sınayıcıdaki optimizasyonu sırasında girdi değişkeninin kullanımını belirler: değer,değişim birimi (adım değeri), başlangıç değeri ve son değer	Optimizasyon Sonuçları ile Çalışma

Fonksiyon	Eylem	Bölüm
Period	Mevcut çizelge zaman aralığına dönüş yapar	Durun Kontrolü
PeriodSeconds	Periyottaki saniye sayısına dönüş yapar	Yaygın Fonksiyonlar
PlaySound	Bir ses dosyasını oynatır	Yaygın Fonksiyonlar
PlotIndexGetInteger	tam-sayı tipli gösterge çizgisi özelliğinin değerine dönüş yapar	Özel Göstergeler
PlotIndexSetDouble	double tipi gösterge çizgisi özelliğinin değerini ayarlar	Özel Göstergeler
PlotIndexSetInteger	int tipi gösterge çizgisi özelliğinin değerini ayarlar	Özel Göstergeler
PlotIndexSetString	string tipi gösterge çizgisi özelliğinin değerini ayarlar	Özel Göstergeler
Point	Mevcut sembolün karşıt dövizindeki puan büyüklüğüne dönüş yapar	Durun Kontrolü
PositionGetDouble	Açık bir pozisyonun istenen (double) özelliğine dönüş yapar	Alım-Satım Fonksiyonları
PositionGetInteger	Açık bir pozisyonun istenen (datetime veya int) özelliğine dönüş yapar	Alım-Satım Fonksiyonları
PositionGetString	Açık bir pozisyonun istenen (string) özelliğine dönüş yapar	Alım-Satım Fonksiyonları
PositionGetSymbol	Açık pozisyona karşılık gelen sembol ismine dönüş yapar	Alım-Satım Fonksiyonları
PositionSelect	Bir açık pozisyonu daha sonraki çalışmalar için seçer	Alım-Satım Fonksiyonları
PositionsTotal	Açık pozisyonların sayısına dönüş yapar	Alım-Satım Fonksiyonları
pow	Bir sayının kuvvetini alır	Matematik Fonksiyonları
Print	Günlük içinde bir mesaj görüntüler	Yaygın Fonksiyonlar
PrintFormat	Önceden ayarlanmış biçime uygun olarak, günlük içindeki sembol kümelerini biçimlendirip çıktılar	Yaygın Fonksiyonlar
rand	0 ile 32767 arasında bir (psuedo) rassal sayıya dönüş yapar	Matematik Fonksiyonları

Fonksiyon	Eylem	Bölüm
ResetLastError	Önceden belirlenmiş _LastError değerini sıfır yapar	Yaygın Fonksiyonlar
ResourceCreate	Bir veri setinin temelinde, bir görüntü kaynağı oluşturur	Yaygın Fonksiyonlar
ResourceFree	Dinamik olarak oluşturulmuş kaynağı siler, (bunun için ayrılmış belleği serbest bırakarak)	Yaygın Fonksiyonlar
ResourceReadImage	ResourceCreate() fonksiyonuyla oluşturulan veya derleme sırasında EX5 dosyasına kaydedilen grafiksel kaynaktan veri okur	Yaygın Fonksiyonlar
ResourceSave	Bir kaynağı belirtilen bir dosya içine kaydeder	Yaygın Fonksiyonlar
round	Değeri, en yakın tamsayıya yuvarlar	Matematik Fonksiyonları
SendFTP	"FTP" sekmesinin ayarlar penceresinde belirtilen adrese bir dosya gönderir	Yaygın Fonksiyonlar
SendMail	"Email" sekmesinin ayarlar penceresinde belirtilen adrese bir dosya gönderir	Yaygın Fonksiyonlar
SendNotification	MetaQuotes ID'si "Notifications" (bildirimler) sekmesinde belirlenen mobil terminallere anlık bildirimler gönderir	Yaygın Fonksiyonlar
SeriesInfoInteger	Tarihsel verinin durumu ile ilgili veriye dönüş yapar	Zaman Serilerine ve Göstergelere Erişim
SetIndexBuffer	Belirtilen gösterge tamponunu, tek boyutlu, double tipli dinamik bir diziye bağlar	Özel Göstergeler
ShortArrayToString	Dizinin bir bölümünü, dizgi içine kopyalar	Dönüşüm Fonksiyonları
ShortToString	Sembol kodunu (unicode), tek sembollük bir dizgiye dönüştürür	Dönüşüm Fonksiyonları
SignalBaseGetDouble	Returns the value of double type property for selected signal	Trade Signals
SignalBaseGetInteger	Returns the value of integer type property for selected signal	Trade Signals

Fonksiyon	Eylem	Bölüm
SignalBaseGetString	Returns the value of string type property for selected signal	Trade Signals
SignalBaseSelect	Selects a signal from signals, available in terminal for further working with it	Trade Signals
SignalBaseTotal	Returns the total amount of signals, available in terminal	Trade Signals
SignalInfoGetDouble	Returns the value of double type property of signal copy settings	Trade Signals
SignalInfoGetInteger	Returns the value of integer type property of signal copy settings	Trade Signals
SignalInfoGetString	Returns the value of string type property of signal copy settings	Trade Signals
SignalInfoSetDouble	Sets the value of double type property of signal copy settings	Trade Signals
SignalInfoSetInteger	Sets the value of integer type property of signal copy settings	Trade Signals
SignalSubscribe	Subscribes to the trading signal	Trade Signals
SignalUnsubscribe	Cancels subscription	Trade Signals
sin	Bir sayının sinüsüne dönüş yapar	Matematik Fonksiyonları
Sleep	Mevcut Uzman Danışmanın veya betiğin çalışmasını, belirlenen bir aralık boyunca bekletir	Yaygın Fonksiyonlar
SocketCreate	Belirtilen bayraklara sahip bir soket oluştur ve onun tanıtıcı değerini geri döndür	Ağ fonksiyonları
SocketClose	Bir soketi kapat	Ağ fonksiyonları
SocketConnect	Zaman aşımı kontrolü ile sunucuya bağlan	Ağ fonksiyonları
SocketIsConnected	Soketin şu anda bağlı olup olmadığını kontrol et	Ağ fonksiyonları
SocketIsReadable	Bir soketten okunabilen bir dizi baytı al	Ağ fonksiyonları
SocketIsWritable	Verilerin şu anda bir sokete yazılıp yazılmayacağını kontrol et	Ağ fonksiyonları
SocketTimeouts	Soket sistem nesnesi için veri almak ve göndermek için zaman aşımını ayarla	Ağ fonksiyonları

Fonksiyon	Eylem	Bölüm
SocketRead	Bir soketten veri oku	Ağ fonksiyonları
SocketSend	Bir sokete veri yaz	Ağ fonksiyonları
SocketTlsHandshake	TLS Tokalaşma protokolü aracılığıyla belirli bir ana bilgisayara güvenli TLS (SSL) bağlantısı kur	Ağ fonksiyonları
SocketTlsCertificate	Ağ bağlantısını güvenli hale getirmek için kullanılan sertifika hakkında veri al	Ağ fonksiyonları
SocketTlsRead	Verileri güvenli TLS bağlantısından oku	Ağ fonksiyonları
SocketTlsReadAvailable	Kullanılabilir tüm verileri güvenli TLS bağlantısından oku	Ağ fonksiyonları
SocketTlsSend	Verileri güvenli TLS bağlantısıyla gönder	Ağ fonksiyonları
sqrt	Sayının karekköküne dönüş yapar	Matematik Fonksiyonları
srand	Bir dizi (psuedo) rassal sayı oluşturmak için başlangıç değeri ayarlar	Matematik Fonksiyonları
StringAdd	Bir dizgiyi başka bir dizginin sonuna ekler	Dizgi Fonksiyonları
StringBufferLen	Dizgi için tahsis edilmiş olan tamponun (ara-bellek) boyutuna dönüş yapar	Dizgi Fonksiyonları
StringCompare	İki dizgiyi karşılaştırır. Birinci dizgi ikincisinden büyük ise 1; dizgiler eşitse 0; ilk dizgi ikincisinden küçükse -1 dönüşü yapar.	Dizgi Fonksiyonları
StringConcatenate	Geçirilen parametrelerden oluşan bir dizgiyi şekillendirir	Dizgi Fonksiyonları
StringFill	Belirtilen bir dizgiyi, seçilen sembollerle doldurur	Dizgi Fonksiyonları
StringFind	Dizgi içinde, bir alt-dizgiyi arar	Dizgi Fonksiyonları
StringFormat	Önceden ayarlanmış biçime göre, sayıyı dizgiye dönüştürür	Dönüşüm Fonksiyonları
StringGetCharacter	Dizgi içerisinde, belirtilen konumdaki bir sayının değerine dönüş yapar	Dizgi Fonksiyonları

Fonksiyon	Eylem	Bölüm
StringInit	Belirtilen sembollerle bir dizgiyi başlatır ve belirtilen dizgi uzunluğunu verir	Dizgi Fonksiyonları
StringLen	Bir dizgi içindeki sembollerin sayısını verir	Dizgi Fonksiyonları
StringReplace	Dizgi içinde bulunan belirli alt-dizgileri, belirtilen sembol dizileriyle değiştirir	Dizgi Fonksiyonları
StringSetCharacter	Belirtilen konumundaki sembol değeri değiştirilmiş bir dizginin kopyasına dönüş yapar	Dizgi Fonksiyonları
StringSplit	Belirtilen ayracı kullanarak, belirtilen bir dizgiden alt-dizgiler elde eder ve bu alt-dizgilerin sayısına dönüş yapar	Dizgi Fonksiyonları
StringSubstr	Bir metnin belirtilen konumundan bir alt-dizgiyi ayıklar	Dizgi Fonksiyonları
StringToCharArray	Unicode'dan ANSI'ye dönüştürülmüş bir dizgiyi, uchar tipi bir dizinin seçilen kısmına sembollerle kopyalar	Dönüşüm Fonksiyonları
StringToColor	Bir "R,G,B" dizgisini veya renk ismini içeren bir dizgiyi, color tipli bir değere dönüştürür	Dönüşüm Fonksiyonları
StringToDouble	Bir sayının sembol temsilini içeren bir dizgiyi, double tipli bir sayıya dönüştürür	Dönüşüm Fonksiyonları
StringToInteger	Bir sayının sembol temsilini içeren bir dizgiyi, int tipli bir sayıya dönüştürür	Dönüşüm Fonksiyonları
StringToLower	Bir dizginin tüm sembollerini, konuma göre küçük karaktere çevirir	Dizgi Fonksiyonları
StringToShortArray	Bir dizgiyi, ushort tipi bir dizinin seçilen kısmına sembollerle kopyalar	Dönüşüm Fonksiyonları
StringToTime	Bir zamanı veya tarihi "yyyy.mm.dd [hh:mi]" biçiminde içeren bir dizgiyi, datetime tipine dönüştürür	Dönüşüm Fonksiyonları

Fonksiyon	Eylem	Bölüm
StringToUpper	Bir dizginin tüm sembollerini, konuma göre büyük karakterlere çevirir	Dizgi Fonksiyonları
StringTrimLeft	Dizginin sol tarafındaki satır atlama karakterlerini, boşlukları ve sekmeleri kırpar	Dizgi Fonksiyonları
StringTrimRight	Dizginin sağ tarafındaki satır atlama karakterlerini, boşlukları ve sekmeleri keser	Dizgi Fonksiyonları
StructToTime	MqlDateTime yapı tipindeki bir değişkenin tipini datetime tipine dönüştürür	Tarih ve Zaman
Symbol	Mevcut çizelgenin sembol ismine dönüş yapar	Durun Kontrolü
SymbolInfoDouble	Karşılık gelen özellik için, sembolün double tipli değerine dönüş yapar	Piyasa Bilgisi
SymbolInfoInteger	Karşılık gelen özellik için, belirtilen sembolün tam-sayı tipi (long, datetime, int veya bool) değerine dönüş yapar	Piyasa Bilgisi
SymbolInfoMarginRate	Returns the margin rates depending on the order type and direction	Piyasa Bilgisi
SymbolInfoSessionQuote	Belirtilen sembol ve gün için, belirtilen fiyatlama seanslarının başlangıç ve bitiş zamanlarının alınmasını sağlar.	Piyasa Bilgisi
SymbolInfoSessionTrade	Belirtilen sembol ve gün için, belirtilen alım-satım seanslarının başlangıç ve bitiş zamanlarının alınmasını sağlar.	Piyasa Bilgisi
SymbolInfoString	Karşılık gelen özellik için, belirtilen sembolün string tipli değerine dönüş yapar	Piyasa Bilgisi
SymbolInfoTick	MqlTick tipi bir değişken ile, belirtilen sembol için mevcut fiyat değerlerine dönüş yapar	Piyasa Bilgisi
SymbolIsSynchronized	Seçilen bir sembole dair terminalde bulunan veri ile alım satım sunucusundaki verinin	Piyasa Bilgisi

Fonksiyon	Eylem	Bölüm
	senkronize olup olmadığını denetler	
SymbolName	Belirtilen sembolün ismine dönüş yapar	Piyasa Bilgisi
SymbolSelect	Piyasa Gözlemi penceresindeki bir sembolü seçer veya bir sembolü pencereden siler	Piyasa Bilgisi
SymbolsTotal	Mevcut sembollerin sayısına (Piyasa Gözleminde seçilmiş olanlara veya hepsine) dönüş yapar	Piyasa Bilgisi
tan	Bir sayının tanjantını alır	Matematik Fonksiyonları
TerminalClose	Terminale işlemi tamamlama komutu verir	Yaygın Fonksiyonlar
TerminalInfoDouble	Programın ortamının karşılık gelen özelliğinin double tipli değerine dönüş yapar	Durun Kontrolü
TerminalInfoInteger	Programın ortamının karşılık gelen özelliğinin tamsayı değerine dönüş yapar	Durun Kontrolü
TerminalInfoString	Programın ortamının karşılık gelen özelliğinin string tipli değerine dönüş yapar	Durun Kontrolü
TesterStatistics	Sınama sonuçlarına dayanarak hesaplanan belirli bir istatistiğin değerine dönüş yapar	Yaygın Fonksiyonlar
TextGetSize	Mevcut yazı tipi ayarları için dizginin genişlik ve yükseklik değerlerine dönüş yapar	Nesne Fonksiyonları
TextOut	Bir metni, grafiksel kaynak oluşturmak amacıyla tasarlanmış özel bir diziye (tampona) aktarır	Nesne Fonksiyonları
TextSetFont	Çizim yöntemlerini kullanarak, görüntülenecek yazı tipini ayarlar (varsayılan olarak Arial 20 kullanılır)	Nesne Fonksiyonları
TimeCurrent	Bilinen son sunucu zamanına (son kotasyonun alındığı zamana) datetime biçiminde dönüş yapar	Tarih ve Zaman

Fonksiyon	Eylem	Bölüm
TimeDaylightSavings	Yaz Saati Uygulamasının değişim işaretine dönüş yapar	Tarih ve Zaman
TimeGMT	Müşteri terminalinin çalışmakta olduğu bilgisayarın yerel zamanını kullanarak, Yaz Saati Uygulaması ile, GMT değerine datetime biçiminde dönüş yapar	Tarih ve Zaman
TimeGMTOffset	GMT zamanıyla yerel bilgisayar zamanı arasındaki farka, saniye bazında ve Yaz Saati Uygulamasını göz önünde bulundurarak dönüş yapar	Tarih ve Zaman
TimeLocal	Yerel bilgisayar zamanına datetime biçiminde dönüş yapar	Tarih ve Zaman
TimeToString	01.01.1970 tarihinden bu yana geçen saniyeleri içeren bir değeri, "yyyy.mm.dd hh:mi" biçiminde bir dizgiye dönüştürür	Dönüşüm Fonksiyonları
TimeToStruct	Bir datetime değişkeninin değerini MqlDateTime yapı tipine dönüştürür	Tarih ve Zaman
TimeTradeServer	Alım-satım sunucusunun mevcut hesaplanmış zamanına dönüş yapar	Tarih ve Zaman
UninitializeReason	Sonlandırma sebebi koduna dönüş yapar	Durun Kontrolü
WebRequest	Sends HTTP request to the specified server	Yaygın Fonksiyonlar
ZeroMemory	Referansla geçirilen bir değeri sıfırlar. Değişken, yapıcılara sahip olan yapı ve sınıflar hariç her tipte olabilir.	Yaygın Fonksiyonlar

MQL5 Sabitlerinin Listesi

Alfabetik sıraya göre tüm MQL5 sabitleri.

Sabit	Açıklama	Usage
__DATE__	Dosya derleme tarihi (zaman olmadan, yani saatler, dakikalar ve saniyeler 0'a eşit olacak şekilde)	Print
__DATETIME__	Dosya derleme tarihi ve zamanı	Print
__FILE__	Derlenmiş mevcut dosyanın ismi	Print
__FUNCSIG__	Makro gövdesinin yerleştirildiği fonksiyonun	Print

Sabit	Açıklama	Usage
	işareti. aşırı-yüklenmiş fonksiyonların tanımlanmasında, fonksiyonun eksiksiz bir tarifi kullanışlı olabilir	
__FUNCTION__	Makronun, gövdesine yerleştirildiği fonksiyonun ismi	Print
__LINE__	Kaynak kodunda makronun yer aldığı satır sayısı	Print
__MQLBUILD__, __MQL5BUILD__	Derleyici sürüm numarası	Print

Sabit	Açıklama	Usage
__PATH__	Derlenmekte olan dosyanın, salt adresi	Print
ACCOUNT_ASSETS	The current assets of an account	AccountInfoDouble
ACCOUNT_BALANCE	Mevduat dövizi cinsinden hesap bakiyesi	AccountInfoDouble
ACCOUNT_COMMISSION_BLOCKED	The current blocked commission amount on an account	AccountInfoDouble
ACCOUNT_COMPANY	Hesaba hizmet veren firmanın ismi	AccountInfoString
ACCOUNT_CREDIT	Mevduat dövizi	AccountInfoDouble

Sabit	Açıklama	Usage
	cinsinden hesap kredisi	
ACCOUNT_CURRENCY	Mevduat dövizinin cinsi	AccountInfoString
ACCOUNT_EQUITY	Mevduat dövizi cinsinden hesaptaki net varlık değeri	AccountInfoDouble
ACCOUNT_LEVERAGE	Hesap kaldıracı	AccountInfoInteger
ACCOUNT_LIABILITIES	The current liabilities on an account	AccountInfoDouble
ACCOUNT_LIMIT_ORDERS	İzin verilen maksimum bekleyen emir sayısı	AccountInfoInteger
ACCOUNT_LOGIN	Hesap numarası	AccountInfoInteger

Sabit	Açıklama	Usage
ACCOUNT_MARGIN	Mevduat döviz birimi cinsinden kullanılan teminat (teminat)	AccountInfoDouble
ACCOUNT_MARGIN_FREE	Mevduat döviz birimi cinsinden serbest teminat	AccountInfoDouble
ACCOUNT_MARGIN_INITIAL	Initial margin. The amount reserved on an account to cover the margin of all pending orders	AccountInfoDouble
ACCOUNT_MARGIN_LEVEL	Yüzde olarak, hesap temin	AccountInfoDouble

Sabit	Açıklama	Usage
	at seviyesi	
ACCOUNT_MARGIN_MAINTENANCE	Maintenance margin. The minimum equity reserved on an account to cover the minimum amount of all open positions	AccountInfoDouble
ACCOUNT_MARGIN_SO_CALL	Marjin çağrısı seviyesi. ACCOUNT_MARGIN_SO_MODE'a bağlı olarak, yüzdelik şekilde	AccountInfoDouble

Sabit	Açıklama	Usage
	veya hesap para birimi cinsiyle ifade edilir	
ACCOUNT_MARGIN_SO_MODE	İzin verilen en küçük teminat için ayar modu	AccountInfoInteger
ACCOUNT_MARGIN_SO_SO	Marjin StopOut seviyesi. ACCOUNT_MARGIN_SO_MODE'a bağlı olarak, yüzdelik şekilde veya hesap para birimi cinsiyle ifade edilir	AccountInfoDouble
ACCOUNT_NAME	Müşteri ismi	AccountInfoString

Sabit	Açıklama	Usage
ACCOUNT_PROFIT	Bir hesabın mevduat dövizi cinsinden mevcut karı	AccountInfoDouble
ACCOUNT_SERVER	Alım-satım sunucusunun ismi	AccountInfoString
ACCOUNT_STOPOUT_MODE_MONEY	Parasal olarak hesap StopOut modu	AccountInfoInteger
ACCOUNT_STOPOUT_MODE_PERCENT	Yüzdelik olarak hesap StopOut modu	AccountInfoInteger
ACCOUNT_TRADE_ALLOWED	Mevcut hesap için izin erilen alım-satım	AccountInfoInteger
ACCOUNT_TRADE_EXPERT	Bir Uzman Danışman için izin	AccountInfoInteger

Sabit	Açıklama	Usage
	verilen alım-satım	
ACCOUNT_TRADE_MODE	Hesap alım-satım modu	AccountInfoInteger
ACCOUNT_TRADE_MODE_CONTEST	Yarışma hesabı	AccountInfoInteger
ACCOUNT_TRADE_MODE_DEMO	Deneme (demo) hesabı	AccountInfoInteger
ACCOUNT_TRADE_MODE_REAL	Gerçek hesap	AccountInfoInteger
ALIGN_CENTER	Ortalanmış (sadece Edit nesnesi için)	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ALIGN_LEFT	Sola hizalama	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ALIGN_RIGHT	Sağa hizalama	ObjectSetInteger, ObjectGetInteger, ChartScreenShot
ANCHOR_CENTER	Nesnenin tam ortasındaki tuttur ma	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	noktası	
ANCHOR_LEFT	Merkezden sola doğru tuttur ma noktası	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT_LOWER	Sol alt köşedeki tuttur ma noktası	ObjectSetInteger, ObjectGetInteger
ANCHOR_LEFT_UPPER	Sol üst köşedeki tuttur ma noktası koordinatı	ObjectSetInteger, ObjectGetInteger
ANCHOR_LOWER	Merkezin aşağısındaki tuttur ma noktası	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT	Merkezden sağa doğru tuttur ma noktası	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
ANCHOR_RIGHT_LOWER	Alt sağ köşedeki tuttur ma nokta sı	ObjectSetInteger, ObjectGetInteger
ANCHOR_RIGHT_UPPER	Sağ üst köşedeki tuttur ma nokta sı	ObjectSetInteger, ObjectGetInteger
ANCHOR_UPPER	Merkezin üstündeki tuttur ma nokta sı	ObjectSetInteger, ObjectGetInteger
BASE_LINE	Ana çizgi	Gösterge Çizgileri
BOOK_TYPE_BUY	Alış Emri	MqlBookInfo
BOOK_TYPE_BUY_MARKET	Piyasa fiyatından alış emri	MqlBookInfo
BOOK_TYPE_SELL	Satış Emri	MqlBookInfo
BOOK_TYPE_SELL_MARKET	Piyasa fiyatından satış emri	MqlBookInfo
BORDER_FLAT	Düz biçim	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
BORDER_RAISED	Çıkıntılı biçim	ObjectSetInteger, ObjectGetInteger
BORDER_SUNKEN	Konkav biçim	ObjectSetInteger, ObjectGetInteger
CHAR_MAX	char tipi ile temsil edilebilecek maksimal değer	Nümerik Tip Sabitleri
CHAR_MIN	char tipi ile temsil edilebilecek minimal değer	Nümerik Tip Sabitleri
CHART_AUTOSCROLL	Çizelgenin sağ sınırına otomatik hareket etme modu	ChartSetInteger, ChartGetInteger
CHART_BARS	Çubuk dizili mi şeklinde gösterir	ChartSetInteger
CHART_BEGIN	Çizelge	ChartNavigate

Sabit	Açıklama	Usage
	başlangıcı (en eski fiyat)	
CHART_BRING_TO_TOP	Çizelgeyi diğer çizelgelerin üstünde göster	ChartSetInteger, ChartGetInteger
CHART_CANDLES	Japon mumları şeklinde gösterir	ChartSetInteger
CHART_COLOR_ASK	Satış fiyatı seviyesi rengi	ChartSetInteger, ChartGetInteger
CHART_COLOR_BACKGROUND	Çizelge arka plan rengi	ChartSetInteger, ChartGetInteger
CHART_COLOR_BID	Satış fiyatı seviyesi rengi	ChartSetInteger, ChartGetInteger
CHART_COLOR_CANDLE_BEAR	Ayı mumunun gövde rengi	ChartSetInteger, ChartGetInteger
CHART_COLOR_CANDLE_BULL	Boğa mumunun	ChartSetInteger, ChartGetInteger

Sabit	Açıklama	Usage
	gövde rengi	
CHART_COLOR_CHART_DOWN	Düşüş çubuğunun, ayı mumunun gölgesinin ve gövde sınırlarının rengi	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_LINE	Çizgi çizelgesi rengi ve Japon "Doji" mumları	ChartSetInteger, ChartGetInteger
CHART_COLOR_CHART_UP	Yükseliş çubuğunun, boğa mumunun gölgesinin ve gövde sınırlarının rengi	ChartSetInteger, ChartGetInteger
CHART_COLOR_FOREGROUND	Eksenlerin, ölçeklerin ve OHLC çizgisi	ChartSetInteger, ChartGetInteger

Sabit	Açıklama	Usage
	nin rengi	
CHART_COLOR_GRID	Izgara rengi	ChartSetInteger, ChartGetInteger
CHART_COLOR_LAST	Son gerçekleşen fiyatın (Last) rengi	ChartSetInteger, ChartGetInteger
CHART_COLOR_STOP_LEVEL	Stop (durdurma) emri seviyelerinin rengi (Zarar Durdur ve Kar Al)	ChartSetInteger, ChartGetInteger
CHART_COLOR_VOLUME	Hacimlerin ve pozisyon açma seviyelerinin rengi	ChartSetInteger, ChartGetInteger
CHART_COMMENT	Çizelgedeki yorumun metni	ChartSetString, ChartGetString
CHART_CURRENT_POS	Şimdiki konum	ChartNavigate
CHART_DRAG_TRADE_LEVELS	Alım-satım	ChartSetInteger, ChartGetInteger

Sabit	Açıklama	Usage
	seviyelerinin çizelge üzerinde fare ile sürüklenmesi izni. Sürükleme modu varsayılan olarak devre dedir (true değeri)	
CHART_EVENT_MOUSE_MOVE	Yapılan fare hareketleri ni ve tıklamalarını (CHARTEVENT_MOUSE_MOVE) çizelge üzerindeki tüm MQL5 programlarına gönder	ChartSetInteger, ChartGetInteger

Sabit	Açıklama	Usage
CHART_EVENT_OBJECT_CREATE	Yeni nesne oluşturulması olayının bildirilerini (CHARTEVENT_OBJECT_CREATE), çizelge üzerindeki tüm MQL5 programlarına gönder	ChartSetInteger, ChartGetInteger
CHART_EVENT_OBJECT_DELETE	Nesne silinmesi olayının bildirilerini (CHARTEVENT_OBJECT_DELETE), çizelge üzerindeki tüm MQL5 programları	ChartSetInteger, ChartGetInteger

Sabit	Açıklama	Usage
	na gönder	
CHART_FIRST_VISIBLE_BAR	Çizelgedeki, görünür ilk çubuğun numarası. Çubukların indislenmesi zaman serilerinde olduğu gibidir.	ChartSetInteger, ChartGetInteger
CHART_FIXED_MAX	Sabitlenmiş çizelge maksimumu	ChartSetDouble, ChartGetDouble
CHART_FIXED_MIN	Sabitlenmiş çizelge minimumu	ChartSetDouble, ChartGetDouble
CHART_FIXED_POSITION	yüzdelik değerle, sol sınırdan sabit çizelge	ChartSetDouble, ChartGetDouble

Sabit	Açıklama	Usage
	konumu. Sabit çizelge pozisyonu, yatay zaman ekseni üzerinde küçük gri bir üçgen ile imlenmiştir. Bu üçgen sadece, yeni tik anında çizelgenin otomatik kaydırılması devre dışı bırakılmışsa görüntülenir (CHART_AUTOSCROLL özelliğine	

Sabit	Açıklama	Usage
	bakın). Bir sabit pozisyon üzerindeki çubuk, yakınlaştırma ve uzaklaştırma sırasında aynı yerde kalır.	
CHART_FOREGROUND	Ön plandaki fiyat çizelgesi	ChartSetInteger, ChartGetInteger
CHART_HEIGHT_IN_PIXELS	Piksel bazında çizelge yüksekliği	ChartSetInteger, ChartGetInteger
CHART_IS_OBJECT	"Çizelge" (OBJ_CHART) nesnesini tanımlama - bir grafiksel nesne için	ChartSetInteger, ChartGetInteger

Sabit	Açıklama	Usage
	'true' değerine döner. gerçek bir çizelge için false dönüşü yapar	
CHART_LINE	Kapanış fiyatları ile çizilen bir çizgi şeklinde gösterir	ChartSetInteger
CHART_MODE	Çizelge tipi (mumlar, çubuklar veya çizgiler)	ChartSetInteger, ChartGetInteger
CHART_MOUSE_SCROLL	Sol fare tuşunu kullanarak çizelgeyi yatay olarak kaydırma. Eğer	ChartSetInteger, ChartGetInteger

Sabit	Açıklama	Usage
	şu parametreler 'true' olarak ayarlanmışsa dikey kaydırma da mümkündür: CHART_SCALEFIX, CHART_SCALEFIX_11 veya CHART_SCALE_PT_PER_BAR	
CHART_POINTS_PER_BAR	Çubuk başına düşen puan bazında ölçek	ChartSetDouble, ChartGetDouble
CHART_PRICE_MAX	Çizelge maksimumu	ChartSetDouble, ChartGetDouble
CHART_PRICE_MIN	Çizelge minimumu	ChartSetDouble, ChartGetDouble
CHART_SCALE	Ölçek	ChartSetInteger, ChartGetInteger

Sabit	Açıklama	Usage
CHART_SCALE_PT_PER_BAR	Çubuk başına düşen puan bazında belirlenen ölçek	ChartSetInteger, ChartGetInteger
CHART_SCALEFIX	Sabit ölçek modu	ChartSetInteger, ChartGetInteger
CHART_SCALEFIX_11	1:1 ölçek modu	ChartSetInteger, ChartGetInteger
CHART_SHIFT	Fiyat çizelgesinin sağ sınırdan girintisinin modu	ChartSetInteger, ChartGetInteger
CHART_SHIFT_SIZE	Sıfır çubuğunun sağ sınırdan olan girintisinin yüzdelik değeri	ChartSetDouble, ChartGetDouble
CHART_SHOW_ASK_LINE	Alış değerlerini çizelgede bir yatay çizgi	ChartSetInteger, ChartGetInteger

Sabit	Açıklama	Usage
	olarak görüntüle	
CHART_SHOW_BID_LINE	Satış değerlerini çizelgede bir yatay çizgi olarak görüntüle	ChartSetInteger, ChartGetInteger
CHART_SHOW_DATE_SCALE	Çizelge üzerinde zaman ölçeğinin gösterilmesi	ChartSetInteger, ChartGetInteger
CHART_SHOW_GRID	Çizelgede ızgarayı görüntüle	ChartSetInteger, ChartGetInteger
CHART_SHOW_LAST_LINE	Son değerleri çizelgede bir yatay çizgi olarak görüntüle	ChartSetInteger, ChartGetInteger
CHART_SHOW_OBJECT_DESCR	Grafiksel nesnelerin	ChartSetInteger, ChartGetInteger

Sabit	Açıklama	Usage
	kendiliğinden açılan (pop-up) tarifleri	
CHART_SHOW_OHLC	OHLC değerlerini sağ üst köşede göster	ChartSetInteger, ChartGetInteger
CHART_SHOW_ONE_CLICK	Showing the "One click trading" panel on a chart	ChartSetInteger, ChartGetInteger
CHART_SHOW_PERIOD_SEP	Bitişik periyotlar arsında dikey ayraçlar görüntüle	ChartSetInteger, ChartGetInteger
CHART_SHOW_PRICE_SCALE	Çizelge üzerinde fiyat ölçeğinin gösterilmesi	ChartSetInteger, ChartGetInteger

Sabit	Açıklama	Usage
CHART_SHOW_TRADE_LEVELS	Alım-satım seviyelerinin çizelgede gösterilmesi (açık pozisyonların, Stop Loss, Take Profit ve bekleyen emirlerin seviyeleri)	ChartSetInteger, ChartGetInteger
CHART_SHOW_VOLUMES	Çizelgede hacimi görüntüle	ChartSetInteger, ChartGetInteger
CHART_VISIBLE_BARS	Çizelge üstündeki görüntülenebilir çubukların sayısı	ChartSetInteger, ChartGetInteger
CHART_VOLUME_HIDE	Hacimler gösterilmez	ChartSetInteger
CHART_VOLUME_REAL	Alım-satım	ChartSetInteger

Sabit	Açıklama	Usage
	hacimleri	
CHART_VOLUME_TICK	Tik Hacimleri	ChartSetInteger
CHART_WIDTH_IN_BARS	Çubuk sayısıyla çizelge genişliği	ChartSetInteger, ChartGetInteger
CHART_WIDTH_IN_PIXELS	Piksel bazında çizelge genişliği	ChartSetInteger, ChartGetInteger
CHART_WINDOW_HANDLE	Çizelge pencere resi işleyicisi (HWND)	ChartSetInteger, ChartGetInteger
CHART_WINDOW_IS_VISIBLE	Alt pencerelerin görünürlüğü	ChartSetInteger, ChartGetInteger
CHART_WINDOW_YDISTANCE	Dikey Y ekseni üzerinde piksel bazında, gösterge alt	ChartSetInteger, ChartGetInteger

Sabit	Açıklama	Usage
	pencere sinin üst çerçe vesi ile, ana çizelge üst çerçe vesi arasın daki uzaklı k. Bir fare olayı durum unda, imleç koordi natlar ı ana çizelg e pence resini n koordi natlar ı şeklin de geçiril ir; göster ge alt pence releri ndeki grafik sel nesnel erin koordi natlar ı ise, alt	

Sabit	Açıklama	Usage
	pencerenin sol üst köşesine göre ayarlanır. Bu değer, koordinatları alt pencerenin sol üst köşesine göre ayarlanan grafiksel nesnelerle doğru şekilde çalışmak amacıyla, ana çizelgenin kesin koordinatlarını alt pencerenin yerel koordinatlarına	

Sabit	Açıklama	Usage
	dönüştürmek için gerekli idir.	
CHART_WINDOWS_TOTAL	Göstergealt pencereleride dahil olmak üzere, toplam çizelge pencere sayısı	ChartSetInteger, ChartGetInteger
CHARTEVENT_CHART_CHANGE	Özellikler iletişim pencere si ile çizelge boyutunun veya çizelge özelliklerinin değişmesi	OnChartEvent
CHARTEVENT_CLICK	Çizelge üstünde tıklama	OnChartEvent

Sabit	Açıklama	Usage
CHARTEVENT_CUSTOM	Bir özel olaylar dizisindeki ilk olayın numarası	OnChartEvent
CHARTEVENT_CUSTOM_LAST	Bir özel olaylar dizisindeki son olayın numarası	OnChartEvent
CHARTEVENT_KEYDOWN	Tuş darbeleri	OnChartEvent
CHARTEVENT_MOUSE_MOVE	Fare hareketi, fare tıklamaları (CHART_EVENT_MOUSE_MOVE=true, çizelge için ayarlanmışsa)	OnChartEvent
CHARTEVENT_OBJECT_CHANGE	Özellikler penceresi	OnChartEvent

Sabit	Açıklama	Usage
	ile değiştirilmiş grafiksel nesne	
CHARTEVENT_OBJECT_CLICK	Grafiksel nesne üzerinde tıklama	OnChartEvent
CHARTEVENT_OBJECT_CREATE	Yaratılmış grafiksel nesne (CHART_EVENT_OBJECT_CREATE=true, çizelge için ayarlanmışsa)	OnChartEvent
CHARTEVENT_OBJECT_DELETE	Silinmiş grafiksel nesne (CHART_EVENT_OBJECT_DELETE=true, çizelge için ayarla	OnChartEvent

Sabit	Açıklama	Usage
	nmışsa)	
CHARTEVENT_OBJECT_DRAG	Grafiksel nesne üzerinde sürükleyip bırakma	OnChartEvent
CHARTEVENT_OBJECT_ENDEDIT	'Düzenle' grafiksel nesnesinde metin düzenlemeişininbitmesi	OnChartEvent
CHARTS_MAX	Terminaldeaynıandaaçıkolabilecekmaksimumçizelgesayısı	Diğer Sabitler
CHIKOUSPAN_LINE	Chiko u Span çizgisi	Gösterge Çizgileri
clrAliceBlue	Alice Blue	Web Renkleri
clrAntiqueWhite	Antiq ue White	Web Renkleri

Sabit	Açıklama	Usage
clrAqua	Aqua	Web Renkleri
clrAquamarine	Aquamarine	Web Renkleri
clrBeige	Beige	Web Renkleri
clrBisque	Bisque	Web Renkleri
clrBlack	Black	Web Renkleri
clrBlanchedAlmond	Blanched Almond	Web Renkleri
clrBlue	Blue	Web Renkleri
clrBlueViolet	Blue Violet	Web Renkleri
clrBrown	Brown	Web Renkleri
clrBurlyWood	Burly Wood	Web Renkleri
clrCadetBlue	Cadet Blue	Web Renkleri
clrChartreuse	Chartreuse	Web Renkleri
clrChocolate	Chocolate	Web Renkleri
clrCoral	Coral	Web Renkleri
clrCornflowerBlue	Cornflower Blue	Web Renkleri
clrCornsilk	Cornsilk	Web Renkleri
clrCrimson	Crimson	Web Renkleri
clrDarkBlue	Dark Blue	Web Renkleri
clrDarkGoldenrod	Dark Goldenrod	Web Renkleri

Sabit	Açıklama	Usage
clrDarkGray	Dark Gray	Web Renkleri
clrDarkGreen	Dark Green	Web Renkleri
clrDarkKhaki	Dark Khaki	Web Renkleri
clrDarkOliveGreen	Dark Olive Green	Web Renkleri
clrDarkOrange	Dark Orange	Web Renkleri
clrDarkOrchid	Dark Orchid	Web Renkleri
clrDarkSalmon	Dark Salmon	Web Renkleri
clrDarkSeaGreen	Dark Sea Green	Web Renkleri
clrDarkSlateBlue	Dark Slate Blue	Web Renkleri
clrDarkSlateGray	Dark Slate Gray	Web Renkleri
clrDarkTurquoise	Dark Turquoise	Web Renkleri
clrDarkViolet	Dark Violet	Web Renkleri
clrDeepPink	Deep Pink	Web Renkleri
clrDeepSkyBlue	Deep Sky Blue	Web Renkleri
clrDimGray	Dim Gray	Web Renkleri

Sabit	Açıklama	Usage
clrDodgerBlue	Dodger Blue	Web Renkleri
clrFireBrick	Fire Brick	Web Renkleri
clrForestGreen	Forest Green	Web Renkleri
clrGainsboro	Gainsboro	Web Renkleri
clrGold	Gold	Web Renkleri
clrGoldenrod	Goldenrod	Web Renkleri
clrGray	Gray	Web Renkleri
clrGreen	Green	Web Renkleri
clrGreenYellow	Green Yellow	Web Renkleri
clrHoneydew	Honeydew	Web Renkleri
clrHotPink	Hot Pink	Web Renkleri
clrIndianRed	Indian Red	Web Renkleri
clrIndigo	Indigo	Web Renkleri
clrIvory	Ivory	Web Renkleri
clrKhaki	Khaki	Web Renkleri
clrLavender	Lavender	Web Renkleri
clrLavenderBlush	Lavender Blush	Web Renkleri
clrLawnGreen	Lawn Green	Web Renkleri
clrLemonChiffon	Lemon Chiffon	Web Renkleri

Sabit	Açıklama	Usage
clrLightBlue	Light Blue	Web Renkleri
clrLightCoral	Light Coral	Web Renkleri
clrLightCyan	Light Cyan	Web Renkleri
clrLightGoldenrod	Light Goldenrod	Web Renkleri
clrLightGray	Light Gray	Web Renkleri
clrLightGreen	Light Green	Web Renkleri
clrLightPink	Light Pink	Web Renkleri
clrLightSalmon	Light Salmon	Web Renkleri
clrLightSeaGreen	Light Sea Green	Web Renkleri
clrLightSkyBlue	Light Sky Blue	Web Renkleri
clrLightSlateGray	Light Slate Gray	Web Renkleri
clrLightSteelBlue	Light Steel Blue	Web Renkleri
clrLightYellow	Light Yellow	Web Renkleri
clrLime	Lime	Web Renkleri
clrLimeGreen	Lime Green	Web Renkleri
clrLinen	Linen	Web Renkleri
clrMagenta	Magenta	Web Renkleri

Sabit	Açıklama	Usage
clrMaroon	Maroon	Web Renkleri
clrMediumAquamarine	Medium Aquamarine	Web Renkleri
clrMediumBlue	Medium Blue	Web Renkleri
clrMediumOrchid	Medium Orchid	Web Renkleri
clrMediumPurple	Medium Purple	Web Renkleri
clrMediumSeaGreen	Medium Sea Green	Web Renkleri
clrMediumSlateBlue	Medium Slate Blue	Web Renkleri
clrMediumSpringGreen	Medium Spring Green	Web Renkleri
clrMediumTurquoise	Medium Turquoise	Web Renkleri
clrMediumVioletRed	Medium Violet Red	Web Renkleri
clrMidnightBlue	Midnight Blue	Web Renkleri
clrMintCream	Mint Crea	Web Renkleri

Sabit	Açıklama	Usage
	m	
clrMistyRose	Misty Rose	Web Renkleri
clrMoccasin	Moccasin	Web Renkleri
clrNavajoWhite	Navajo White	Web Renkleri
clrNavy	Navy	Web Renkleri
clrNONE	Rengin olmaması	Diğer Sabitler
clrOldLace	Old Lace	Web Renkleri
clrOlive	Olive	Web Renkleri
clrOliveDrab	Olive Drab	Web Renkleri
clrOrange	Orange	Web Renkleri
clrOrangeRed	Orange Red	Web Renkleri
clrOrchid	Orchid	Web Renkleri
clrPaleGoldenrod	Pale Goldenrod	Web Renkleri
clrPaleGreen	Pale Green	Web Renkleri
clrPaleTurquoise	Pale Turquoise	Web Renkleri
clrPaleVioletRed	Pale Violet Red	Web Renkleri
clrPapayaWhip	Papaya Whip	Web Renkleri

Sabit	Açıklama	Usage
clrPeachPuff	Peach Puff	Web Renkleri
clrPeru	Peru	Web Renkleri
clrPink	Pink	Web Renkleri
clrPlum	Plum	Web Renkleri
clrPowderBlue	Powder Blue	Web Renkleri
clrPurple	Purple	Web Renkleri
clrRed	Red	Web Renkleri
clrRosyBrown	Rosy Brown	Web Renkleri
clrRoyalBlue	Royal Blue	Web Renkleri
clrSaddleBrown	Saddle Brown	Web Renkleri
clrSalmon	Salmon	Web Renkleri
clrSandyBrown	Sandy Brown	Web Renkleri
clrSeaGreen	Sea Green	Web Renkleri
clrSeashell	Seashell	Web Renkleri
clrSienna	Sienna	Web Renkleri
clrSilver	Silver	Web Renkleri
clrSkyBlue	Sky Blue	Web Renkleri
clrSlateBlue	Slate Blue	Web Renkleri
clrSlateGray	Slate Gray	Web Renkleri
clrSnow	Snow	Web Renkleri

Sabit	Açıklama	Usage
clrSpringGreen	Spring Green	Web Renkleri
clrSteelBlue	Steel Blue	Web Renkleri
clrTan	Tan	Web Renkleri
clrTeal	Teal	Web Renkleri
clrThistle	Thistle	Web Renkleri
clrTomato	Tomato	Web Renkleri
clrTurquoise	Turquoise	Web Renkleri
clrViolet	Violet	Web Renkleri
clrWheat	Wheat	Web Renkleri
clrWhite	White	Web Renkleri
clrWhiteSmoke	White Smoke	Web Renkleri
clrYellow	Yellow	Web Renkleri
clrYellowGreen	Yellow Green	Web Renkleri
CORNER_LEFT_LOWER	Çizelgenin sol alt köşesindeki koordinat merkezi	ObjectSetInteger, ObjectGetInteger
CORNER_LEFT_UPPER	Çizelgenin sol üst köşesindeki koordinat	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	merkezi	
CORNER_RIGHT_LOWER	Çizelgenin sağ alt köşesindeki koordinat merkezi	ObjectSetInteger, ObjectGetInteger
CORNER_RIGHT_UPPER	Çizelgenin sağ üst köşesindeki koordinat merkezi	ObjectSetInteger, ObjectGetInteger
CP_ACP	Mevcut Windows ANSI kod sayfası.	CharArrayToString, StringToCharArray, FileOpen
CP_MACCP	Mevcut sistemin Macintosh kod sayfası. Not: Bu değer, sadece	CharArrayToString, StringToCharArray, FileOpen

Sabit	Açıklama	Usage
	eskiden yazılmış programlarda sıklıkla kullanılır ve artık gerekli değildir; birsüredir Macintosh bilgisayarlarda Unicode kullanılmaktadır	
CP_OEMCP	Mevcut sistemin OEM kod sayfası.	CharArrayToString, StringToCharArray, FileOpen
CP_SYMBOL	Sembol kod sayfası	CharArrayToString, StringToCharArray, FileOpen
CP_THREAD_ACP	Geçerli iş parçacığı için Windo	CharArrayToString, StringToCharArray, FileOpen

Sabit	Açıklama	Usage
	ws ANSI kod sayfası.	
CP_UTF7	UTF-7 kod sayfası.	CharArrayToString, StringToCharArray, FileOpen
CP_UTF8	UTF-8 kod sayfası.	CharArrayToString, StringToCharArray, FileOpen
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

Sabit	Açıklama	Usage
CRYPT_HASH_MD5	MD5 HASH cacula tion	CryptEncode, CryptDecode
CRYPT_HASH_SHA1	SHA1 HASH cacula tion	CryptEncode, CryptDecode
CRYPT_HASH_SHA256	SHA2 56 HASH cacula tion	CryptEncode, CryptDecode
DBL_DIG	doubl e tipi için, anlam lı ondalı k hanel erin sayısı	Nümerik Tip Sabitleri
DBL_EPSILON	Şu koşulu sağlay an minim al değer : 1.0+D BL_EP SILON != 1.0 (doubl e tipi için)	Nümerik Tip Sabitleri
DBL_MANT_DIG	doubl e tipi için manti s içinde	Nümerik Tip Sabitleri

Sabit	Açıklama	Usage
	ki bit sayısı	
DBL_MAX	double tipi ile temsil edilebilecek maksimal değer	Nümerik Tip Sabitleri
DBL_MAX_10_EXP	double tipli maksimal ondalık üs değeri	Nümerik Tip Sabitleri
DBL_MAX_EXP	double tipli maksimal ikili üs değeri	Nümerik Tip Sabitleri
DBL_MIN	double tipi ile temsil edilebilecek minimal pozitif değer	Nümerik Tip Sabitleri
DBL_MIN_10_EXP	double tipli minimal ondalık üs değeri	Nümerik Tip Sabitleri

Sabit	Açıklama	Usage
DBL_MIN_EXP	double tipli minimal ikili üs değeri	Nümerik Tip Sabitleri
DEAL_COMMENT	İşlem yorumu	HistoryDealGetString
DEAL_COMMISSION	İşlem komisyonu	HistoryDealGetDouble
DEAL_ENTRY	Faaliyet yönü - piyasaya giriş, piyasadan çıkış veya dönüş	HistoryDealGetInteger
DEAL_ENTRY_IN	Piyasaya giriş	HistoryDealGetInteger
DEAL_ENTRY_INOUT	U-dönüşü	HistoryDealGetInteger
DEAL_ENTRY_OUT	Piyasadan çıkış	HistoryDealGetInteger
DEAL_MAGIC	Faaliyet için magic number (bakınız, ORDER_MAGIC)	HistoryDealGetInteger

Sabit	Açıklama	Usage
DEAL_ORDER	İşlemin emir numarası	HistoryDealGetInteger
DEAL_POSITION_ID	Pozisyon tanımlayıcısı - işlem içeren açma, değişim veya kapama hallerindeki pozisyon kimliği. Her bir pozisyonun, aktif olduğu süre içinde, sembol için gerçekleştirilmiş tüm işlemlere atanan benzersiz bir tanımlayıcısı	HistoryDealGetInteger

Sabit	Açıklama	Usage
	(kimliği) vardır.	
DEAL_PRICE	İşlem fiyatı	HistoryDealGetDouble
DEAL_PROFIT	İşlem karı	HistoryDealGetDouble
DEAL_SWAP	Kapanıştaki birikmiş swap	HistoryDealGetDouble
DEAL_SYMBOL	İşlem sembolü	HistoryDealGetString
DEAL_TIME	İşlem zamanı	HistoryDealGetInteger
DEAL_TIME_MSC	01.01.1970 tarihinden beri geçen milisaniyeler şeklinde, işlemin gerçekleştirilme zamanı	HistoryDealGetInteger
DEAL_TYPE	İşlem tipi	HistoryDealGetInteger
DEAL_TYPE_BALANCE	Bakiye	HistoryDealGetInteger
DEAL_TYPE_BONUS	Bonus	HistoryDealGetInteger

Sabit	Açıklama	Usage
DEAL_TYPE_BUY	Alış işlemi	HistoryDealGetInteger
DEAL_TYPE_BUY_CANCELED	İptal edilen alış işlemi. Önceden gerçekleştirilmiş bir alım işleminin iptal edildiği bir durumla karşılaşılabilir. Bu durumda, daha önce gerçekleştirilen işlem (DEAL_TYPE_BUY), DEAL_TYPE_BUY_CANCELED tipine dönüş türülür ve ondan	HistoryDealGetInteger

Sabit	Açıklama	Usage
	elde edilen kar/zarar sıfırlanır. Önceden elde edilen kar/zarar ayrı bir bakiye işlemi ile ücretlendirilir/çekilir	
DEAL_TYPE_CHARGE	Ek ücret	HistoryDealGetInteger
DEAL_TYPE_COMMISSION	Ek komisyon	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_DAILY	Günlük ajans komisyonu	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_AGENT_MONTHLY	Aylık ajans komisyonu	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_DAILY	Günlük komisyon	HistoryDealGetInteger
DEAL_TYPE_COMMISSION_MONTHLY	Aylık komisyon	HistoryDealGetInteger

Sabit	Açıklama	Usage
DEAL_TYPE_CORRECTION	Düzeltme	HistoryDealGetInteger
DEAL_TYPE_CREDIT	Kredi	HistoryDealGetInteger
DEAL_TYPE_INTEREST	Faiz oranı	HistoryDealGetInteger
DEAL_TYPE_SELL	Satış	HistoryDealGetInteger
DEAL_TYPE_SELL_CANCELED	İptal edilen satış işlemi. Önceden gerçekleştirilmiş bir satış işlemlerinin iptal edildiği bir durumla karşılaşılabilir. Bu durumda, daha önce gerçekleştirilmiş işlem (DEAL_TYPE_SELL), DEAL_TYPE_SELL_CANC	HistoryDealGetInteger

Sabit	Açıklama	Usage
	ELED tipine dönüş türülür ve ondan elde edilen kar/zarar sıfırlanır. Önceden elde edilen kar/zarar ayrı bir bakiye işlemi ile ücretlendirilir/çekilir	
DEAL_VOLUME	İşlemin hacmi	HistoryDealGetDouble
DRAW_ARROW	Ok çizimi	Çizim Stilleri
DRAW_BARS	Çubuk dizilimi şeklinde gösterir	Çizim Stilleri
DRAW_CANDLES	Mumların dizilimi şeklin	Çizim Stilleri

Sabit	Açıklama	Usage
	de göster ir	
DRAW_COLOR_ARROW	Çok renkli okların çizimi	Çizim Stilleri
DRAW_COLOR_BARS	Çok renkli çubuklar	Çizim Stilleri
DRAW_COLOR_CANDLES	Çok renkli mumlar	Çizim Stilleri
DRAW_COLOR_HISTOGRAM	Sıfır çizgisinden başlayan çok renkli histogram	Çizim Stilleri
DRAW_COLOR_HISTOGRAM2	İki gösterge tamponunun çok renkli histogramı	Çizim Stilleri
DRAW_COLOR_LINE	Çok renkli çizgi	Çizim Stilleri
DRAW_COLOR_SECTION	Çok renkli kesit	Çizim Stilleri
DRAW_COLOR_ZIGZAG	Çok renkli	Çizim Stilleri

Sabit	Açıklama	Usage
	ZigZag	
DRAW_FILLING	İki seviye arasının renk ile doldurulması	Çizim Stilleri
DRAW_HISTOGRAM	Sıfır çizgisinden başlayan histogram	Çizim Stilleri
DRAW_HISTOGRAM2	İki gösterge tamponunun histogramı	Çizim Stilleri
DRAW_LINE	Çizgi	Çizim Stilleri
DRAW_NONE	Çizilmez	Çizim Stilleri
DRAW_SECTION	Bölüm	Çizim Stilleri
DRAW_ZIGZAG	Zigzag stili, çubuklar için dikey kesitler oluşturulmasını sağlar	Çizim Stilleri
ELLIOTT_CYCLE	Döngü (Cycle	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	)	
ELLIOTT_GRAND_SUPERCYCLE	Büyük Süper Döngü (Grand Supercycle)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_INTERMEDIATE	Orta (Intermediate)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINOR	İkincil (Minor)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINUETTE	Saniye (Minuette)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_MINUTE	Dakika	ObjectSetInteger, ObjectGetInteger
ELLIOTT_PRIMARY	Birincil	ObjectSetInteger, ObjectGetInteger
ELLIOTT_SUBMINUETTE	Saniye-altı (Subminuette)	ObjectSetInteger, ObjectGetInteger
ELLIOTT_SUPERCYCLE	Süper döngü (Supercycle)	ObjectSetInteger, ObjectGetInteger
EMPTY_VALUE	Bir gösterge tamponu için boş değer	Diğer Sabitler

Sabit	Açıklama	Usage
ERR_ACCOUNT_WRONG_PROPERTY	Hesap özelliğinin tanımlayıcısı yanlış	GetLastError
ERR_ARRAY_BAD_SIZE	İstenen dizi büyüklüğü 2 GB'ı aşıyor	GetLastError
ERR_ARRAY_RESIZE_ERROR	Dizinin yer değişimi için yeterli bellek yok, veya bir statik dizinin büyüklüğünü değiştirme girişimi yapıldı	GetLastError
ERR_BOOKS_CANNOT_ADD	Piyasa Derinliği eklenemiyor	GetLastError
ERR_BOOKS_CANNOT_DELETE	Piyasa Derinliği kaldırılamıyor	GetLastError

Sabit	Açıklama	Usage
ERR_BOOKS_CANNOT_GET	Piyasa Derinliğinden veri alınamıyor	GetLastError
ERR_BOOKS_CANNOT_SUBSCRIBE	Piyasa Derinliğinden veri alımı için abonelik hatası	GetLastError
ERR_BUFFERS_NO_MEMORY	Gösterge tamponlarının dağıtılması için yeterli bellek yok	GetLastError
ERR_BUFFERS_WRONG_INDEX	Gösterge tamponunun indisi yanlış	GetLastError
ERR_CANNOT_CLEAN_DIRECTORY	Klasörün kaldırılması başarısız oldu (büyük olasılıkla bir veya iki	GetLastError

Sabit	Açıklama	Usage
	dosya bloke edilmiş ve kaldırma işlemi başarısız olmuş)	
ERR_CANNOT_DELETE_DIRECTORY	Klasör kaldırılamaz	GetLastError
ERR_CANNOT_DELETE_FILE	Dosya silme hatası	GetLastError
ERR_CANNOT_OPEN_FILE	Dosya açma hatası	GetLastError
ERR_CHAR_ARRAY_ONLY	char tipli bir dizi olmalı	GetLastError
ERR_CHART_CANNOT_CHANGE	Çizelge periyodunu ve sembolünü değiştirme işlemi başarısız oldu	GetLastError
ERR_CHART_CANNOT_CREATE_TIMER	Sayacı başlatma başarısız	GetLastError

Sabit	Açıklama	Usage
ERR_CHART_CANNOT_OPEN	Çizelge açılma hatası	GetLastError
ERR_CHART_INDICATOR_CANNOT_ADD	Çizelgeye gösterge eklerken hata oluştu	GetLastError
ERR_CHART_INDICATOR_CANNOT_DEL	Göstergeyi çizelgeden silerken hata oluştu	GetLastError
ERR_CHART_INDICATOR_NOT_FOUND	Gösterge belirtilen çizelge üzerinde bulunamadı	GetLastError
ERR_CHART_NAVIGATE_FAILED	Çizelge boyunca konumlandırma hatası	GetLastError
ERR_CHART_NO_EXPERT	Çizelgede, olayı işleyebilecek bir	GetLastError

Sabit	Açıklama	Usage
	Uzman Danışman yok	
ERR_CHART_NO_REPLY	Çizelge yanıt vermiyor	GetLastError
ERR_CHART_NOT_FOUND	Çizelge bulunamadı	GetLastError
ERR_CHART_SCREENSHOT_FAILED	Ekran görüntüsü oluşturma hatası	GetLastError
ERR_CHART_TEMPLATE_FAILED	Şablon uygulama hatası	GetLastError
ERR_CHART_WINDOW_NOT_FOUND	Göstergeyi içeren alt pencere bulunamadı	GetLastError
ERR_CHART_WRONG_ID	Yanlış çizelge kimliği (tanımlayıcısı)	GetLastError
ERR_CHART_WRONG_PARAMETER	Çizelge ile	GetLastError

Sabit	Açıklama	Usage
	çalışma fonksiyonu için kullanılan parametrenin hata değeri	
ERR_CHART_WRONG_PROPERTY	Yanlış çizelge özelliği tanımlayıcısı	GetLastError
ERR_CUSTOM_WRONG_PROPERTY	Özel gösterge özeliğinin tanımlayıcısı yanlış	GetLastError
ERR_DIRECTORY_NOT_EXIST	Böyle bir dosya yolu bulunmuyor	GetLastError
ERR_DOUBLE_ARRAY_ONLY	double tipli bir dizi olmalı	GetLastError
ERR_FILE_BINSTRINGSIZE	Dosya ikili biçimde açıldığı için dizgi büyükl	GetLastError

Sabit	Açıklama	Usage
	üğü belirtilmelidir	
ERR_FILE_CACHEBUFFER_ERROR	Okuma için yeterli önbellek alanı yok	GetLastError
ERR_FILE_CANNOT_REWRITE	Dosya yeniden yazılamaz	GetLastError
ERR_FILE_IS_DIRECTORY	Bu bir dosya değil, bir klasör	GetLastError
ERR_FILE_ISNOT_DIRECTORY	Bu bir dosya, klasör değil	GetLastError
ERR_FILE_NOT_EXIST	Dosya mevcut değil	GetLastError
ERR_FILE_NOTBIN	Dosya, ikili (binary) dosya şeklinde açılmalıdır	GetLastError
ERR_FILE_NOTCSV	Dosya, CSV dosyası	GetLastError

Sabit	Açıklama	Usage
	şeklinde açılmalıdır	
ERR_FILE_NOTTOREAD	Dosya okuma amacıyla açılmalıdır	GetLastError
ERR_FILE_NOTTOWRITE	Dosya yazım amacıyla açılmalıdır	GetLastError
ERR_FILE_NOTTXT	Dosya, metin (txt) dosyası şeklinde açılmalıdır	GetLastError
ERR_FILE_NOTTXTORCSV	Dosya, metin (txt) veya CSV dosyası şeklinde açılmalıdır	GetLastError
ERR_FILE_READERROR	Dosya okuma hatası	GetLastError

Sabit	Açıklama	Usage
ERR_FILE_WRITEERROR	Kaynağı dosyaya yazma işlemi başarısız oldu	GetLastError
ERR_FLOAT_ARRAY_ONLY	float tipli bir dizi olmalı	GetLastError
ERR_FTP_SEND_FAILED	Ftp ile dosya gönderimi başarısız oldu	GetLastError
ERR_FUNCTION_NOT_ALLOWED	Sistem fonksiyonunun çağrılmasına izin verilmiyor	GetLastError
ERR_GLOBALVARIABLE_EXISTS	Aynı isimde bir müşteri terminali global değişkeni mevcut	GetLastError

Sabit	Açıklama	Usage
ERR_GLOBALVARIABLE_NOT_FOUND	Müşteri terminalinin global değişkeni bulunamadı	GetLastError
ERR_HISTORY_NOT_FOUND	İstenen geçmiş bulunamadı	GetLastError
ERR_HISTORY_WRONG_PROPERTY	Geçmiş özeliğinin tanımlayıcısı yanlış	GetLastError
ERR_INCOMPATIBLE_ARRAYS	Uyumlu olmayan dizilerin kopyalanması. Dizgi dizisi sadece bir dizgi dizisine kopyalanabilir, aynı şekilde bir sayısa	GetLastError

Sabit	Açıklama	Usage
	l dizide sadece sayısal diziye kopyalanabilir	
ERR_INCOMPATIBLE_FILE	Metin dosyaları dizgi dizileri için olmalıdır, diğer diziler için ikili (binary) dosyalar kullanılmalıdır	GetLastError
ERR_INDICATOR_CANNOT_ADD	Göstergenin çizelgeye uygulanmasında hata	GetLastError
ERR_INDICATOR_CANNOT_APPLY	Bu göstergeye başka bir göstergeye uygula	GetLastError

Sabit	Açıklama	Usage
	namaz	
ERR_INDICATOR_CANNOT_CREATE	Göstergeoluşturulamıyor	GetLastError
ERR_INDICATOR_CUSTOM_NAME	Dizideki ilk parametre, özel göstergenin ismi olmalı	GetLastError
ERR_INDICATOR_DATA_NOT_FOUND	İstenen veri bulunamadı	GetLastError
ERR_INDICATOR_NO_MEMORY	Göstergeyi eklemek için yeterli bellek yok	GetLastError
ERR_INDICATOR_PARAMETER_TYPE	Göstergeoluşturulması sırasında, dizi içinde geçersiz parametre tipi	GetLastError
ERR_INDICATOR_PARAMETERS_MISSING	Göstergede	GetLastError

Sabit	Açıklama	Usage
	hiç parametre yok	
ERR_INDICATOR_UNKNOWN_SYMBOL	Bilinmeyen sembol	GetLastError
ERR_INDICATOR_WRONG_HANDLE	Göstergenin tanıtıcı değeri yanlış	GetLastError
ERR_INDICATOR_WRONG_INDEX	İstenen gösterge tamponunun indisi hatalı	GetLastError
ERR_INDICATOR_WRONG_PARAMETERS	Göstergedeki parametre sayısı yanlış	GetLastError
ERR_INT_ARRAY_ONLY	int tipli bir dizi olmalı	GetLastError
ERR_INTERNAL_ERROR	Beklenmeyen içsel hata	GetLastError
ERR_INVALID_ARRAY	Yanlış tipli veya yanlış	GetLastError

Sabit	Açıklama	Usage
	büyüklükte bir dizi veya hasarlı bir nesne veya bir dinamik dizi	
ERR_INVALID_DATETIME	Geçersiz tarih ve/veya zaman	GetLastError
ERR_INVALID_FILEHANDLE	Bu tanıtıcı değere sahip bir dosya kapatıldı, veya hiç açılmadı	GetLastError
ERR_INVALID_PARAMETER	Sistem fonksiyonunun çağrısında yanlış fonksiyon kullanımı	GetLastError

Sabit	Açıklama	Usage
ERR_INVALID_POINTER	Yanlış işaretçi	GetLastError
ERR_INVALID_POINTER_TYPE	Yanlış tipte işaretçi	GetLastError
ERR_LONG_ARRAY_ONLY	long tipli bir dizi olmalı	GetLastError
ERR_MAIL_SEND_FAILED	Email gönderimi başarısız oldu	GetLastError
ERR_MARKET_LASTTIME_UNKNOWN	Son tik zamanı bilinmiyor (hiç tik yok)	GetLastError
ERR_MARKET_NOT_SELECTED	Sembol, Piyasa Gözleminde seçilmedi	GetLastError
ERR_MARKET_SELECT_ERROR	Sembolün Piyasa Gözleminden silinmesinde hata	GetLastError

Sabit	Açıklama	Usage
ERR_MARKET_UNKNOWN_SYMBOL	Bilinmeyen sembol	GetLastError
ERR_MARKET_WRONG_PROPERTY	Sembol özelliği için yanlış tanımlayıcı	GetLastError
ERR_MQL5_WRONG_PROPERTY	Program özelliğinin tanımlayıcısı yanlış	GetLastError
ERR_NO_STRING_DATE	Dizgide tarih yok	GetLastError
ERR_NOT_ENOUGH_MEMORY	Sistem fonksiyonunu uygulamak için yeterli bellek yok	GetLastError
ERR_NOTIFICATION_SEND_FAILED	Uyarı gönderme başarısız oldu	GetLastError
ERR_NOTIFICATION_TOO_FREQUENT	Uyarı gönderimi çok sık	GetLastError

Sabit	Açıklama	Usage
ERR_NOTIFICATION_WRONG_PARAMETER	Uyarı gönderimi için geçersiz parametre - boş bir dizgi veya NULL değeri SendNotification() fonksiyonuna geçirilmiş	GetLastError
ERR_NOTIFICATION_WRONG_SETTINGS	Terminalde yanlış uyarı ayarları yapılmış (tanımlayıcı belirtilmemiş veya izin durumu ayarlanmamış)	GetLastError
ERR_NOTINITIALIZED_STRING	Başlatılmam	GetLastError

Sabit	Açıklama	Usage
	ış dizgi	
ERR_NUMBER_ARRAYS_ONLY	Nümerik bir dizi olmalı	GetLastError
ERR_OBJECT_ERROR	Grafiksel nesne ile çalışma hatası	GetLastError
ERR_OBJECT_GETDATE_FAILED	Değere karşılık gelen tarih alınamadı	GetLastError
ERR_OBJECT_GETVALUE_FAILED	Tarihe karşılık gelen değer alınamadı	GetLastError
ERR_OBJECT_NOT_FOUND	Grafiksel nesne bulunamadı	GetLastError
ERR_OBJECT_WRONG_PROPERTY	Grafiksel nesne için yanlış tanımlayıcı	GetLastError
ERR_ONEDIM_ARRAYS_ONLY	Tek boyutl	GetLastError

Sabit	Açıklama	Usage
	u bir dizi olmalı	
ERR_OPENCL_BUFFER_CREATE	OpenCL tamponu oluşturma hatası	GetLastError
ERR_OPENCL_CONTEXT_CREATE	OpenCL bağlamı oluşturma hatası	GetLastError
ERR_OPENCL_EXECUTE	OpenCL programı çalışma zamanı hatası	GetLastError
ERR_OPENCL_INTERNAL	OpenCL çalışması sırasında bir hata oluştu	GetLastError
ERR_OPENCL_INVALID_HANDLE	Geçersiz OpenCL işleyici	GetLastError
ERR_OPENCL_KERNEL_CREATE	OpenCL kernel	GetLastError

Sabit	Açıklama	Usage
	i oluşturma hatası	
ERR_OPENCL_NOT_SUPPORTED	OpenCL fonksiyonları bu bilgisayarda desteklenmiyor	GetLastError
ERR_OPENCL_PROGRAM_CREATE	OpenCL programının derlenmesi sırasında hata oluştu	GetLastError
ERR_OPENCL_QUEUE_CREATE	OpenCL içinde bir çalıştırma kuyruğu oluşturma hatası	GetLastError
ERR_OPENCL_SET_KERNEL_PARAMETER	OpenCL kernel için parametrelerin ayarla	GetLastError

Sabit	Açıklama	Usage
	nması sırasında hata oluştu	
ERR_OPENCL_TOO_LONG_KERNEL_NAME	Kernel ismi çok uzun (OpenCL kernel)	GetLastError
ERR_OPENCL_WRONG_BUFFER_OFFSET	OpenCL tamponu içinde geçersiz konum	GetLastError
ERR_OPENCL_WRONG_BUFFER_SIZE	OpenCL tamponu için geçersiz büyüklük	GetLastError
ERR_PLAY_SOUND_FAILED	Ses oynatımı başarısız oldu	GetLastError
ERR_RESOURCE_NAME_DUPLICATED	dynamic ve static kaynakların isimleri	GetLastError

Sabit	Açıklama	Usage
	uyuşuyor	
ERR_RESOURCE_NAME_IS_TOO_LONG	kaynak ismi 63 karakteri aşıyor	GetLastError
ERR_RESOURCE_NOT_FOUND	EX5 içinde bu isimde kaynak bulunamadı	GetLastError
ERR_RESOURCE_UNSUPPORTED_TYPE	Desteklenmeyen kaynak tipi veya kaynağın boyutu 16 Mb'i geçiyor	GetLastError
ERR_SERIES_ARRAY	Zaman serileri kullanılamaz	GetLastError
ERR_SHORT_ARRAY_ONLY	short tipli bir dizi olmalı	GetLastError
ERR_SMALL_ARRAY	Dizi çok	GetLastError

Sabit	Açıklama	Usage
	küçük, başlangıç konumu dizinin dışında	
ERR_SMALL_ASSERIES_ARRAY	Alınan dizi AS_SERIES şeklinde bildirilmiş ve yeterli büyüklükte değil	GetLastError
ERR_STRING_OUT_OF_MEMORY	Dizgi için yeterli bellek yok	GetLastError
ERR_STRING_RESIZE_ERROR	Dizginin yeniden yerleştirilmesi için yeterli bellek yok	GetLastError
ERR_STRING_SMALL_LEN	Dizgi uzunluğu beklenenden az	GetLastError

Sabit	Açıklama	Usage
ERR_STRING_TIME_ERROR	Dizgiyi tarihe çevirme hatası	GetLastError
ERR_STRING_TOO_BIGNUMBER	Sayı çok büyük, ULONG_MAX değerinden fazla	GetLastError
ERR_STRING_UNKNOWNTYPE	Dizgiye dönüşüm yaparken bilinmeyen veri tipi	GetLastError
ERR_STRING_ZEROADDED	Dizginin sonuna 0 eklenmiş, kullanışsız işlem	GetLastError
ERR_STRINGPOS_OUTOFRANGE	Konum, dizginin dışında	GetLastError
ERR_STRUCT_WITHOBJECTS_ORCLASS	Yapı, dizgi nesnelerini	GetLastError

Sabit	Açıklama	Usage
	ve/veya dinamik dizileri ve/veya bu tip nesnelerin yapılarını ve/veya sınıflarını içeriyor	
ERR_SUCCESS	İşlem başarıyla tamamlandı	GetLastError
ERR_TERMINAL_WRONG_PROPERTY	Terminal özelliğinin tanımlayıcısı yanlış	GetLastError
ERR_TOO_LONG_FILENAME	Dosya ismi çok uzun	GetLastError
ERR_TOO_MANY_FILES	64 dosyadan fazlası aynı anda açılamaz	GetLastError

Sabit	Açıklama	Usage
ERR_TOO_MANY_FORMATTERS	Biçim belirteçlerinin sayısı, parametrelerden fazla	GetLastError
ERR_TOO_MANY_PARAMETERS	Parametrelerin sayısı, biçim belirteçlerinden fazla	GetLastError
ERR_TRADE_DEAL_NOT_FOUND	İşlem bulunamadı	GetLastError
ERR_TRADE_DISABLED	Uzman Danışmanlar ile alım-satım yapmak yasaklanmış	GetLastError
ERR_TRADE_ORDER_NOT_FOUND	Emir bulunamadı	GetLastError
ERR_TRADE_POSITION_NOT_FOUND	Pozisyon bulunamadı	GetLastError
ERR_TRADE_SEND_FAILED	Alım-satım isteği	GetLastError

Sabit	Açıklama	Usage
	nin gönderilmesi başarısız oldu	
ERR_TRADE_WRONG_PROPERTY	Alım-satım özelliği tanımlayıcısı yanlış	GetLastError
ERR_USER_ERROR_FIRST	Kullanıcı tanımlı hatalar bu kod ile başlar	GetLastError
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	Dosya yolunu	GetLastError

Sabit	Açıklama	Usage
	n ismi yanlış	
ERR_WRONG_FILEHANDLE	Dosya için yanlış tanıtıcı değer	GetLastError
ERR_WRONG_FILENAME	Geçersiz dosya ismi	GetLastError
ERR_WRONG_FORMATSTRING	Geçersiz biçim dizgisi	GetLastError
ERR_WRONG_INTERNAL_PARAMETER	Müşteri terminali fonksiyonunun çağrısında yanlış parametre kullanımı	GetLastError
ERR_WRONG_STRING_DATE	Dizgideki tarih yanlış	GetLastError
ERR_WRONG_STRING_OBJECT	Hasarlı dizgi nesnesi	GetLastError
ERR_WRONG_STRING_PARAMETER	string tipi hatalı	GetLastError

Sabit	Açıklama	Usage
	parametre	
ERR_WRONG_STRING_TIME	Dizgideki zaman yanlış	GetLastError
ERR_ZEROSIZE_ARRAY	Dizinin büyüklüğü sıfır	GetLastError
FILE_ACCESS_DATE	En son erişim tarihi	FileGetInteger
FILE_ANSI	ANSI tipli dizgiler (bir baytlık semboller). Bu bayrak, FileOpen() fonksiyonunda kullanılır	FileOpen
FILE_BIN	İkili oku/yaz modu (dizgiden dizgiye dönüşüm olmak	FileOpen

Sabit	Açıklama	Usage
	sızın). Bu bayrak, FileOpen() fonksiyonunda kullanılır	
FILE_COMMON	Tüm müşteri terminalleri için olan genel klasörün dosya yolu \Terminal\Common\Files. Bu bayrak, FileOpen(), FileCopy(), FileMove() ve FileIsExist() fonksiyonlarında kullanılır.	FileOpen, FileCopy, FileMove, FileIsExist

Sabit	Açıklama	Usage
FILE_CREATE_DATE	Oluşturulma tarihi	FileGetInteger
FILE_CSV	CSV dosyası (tüm elemanları uygun tipte dizgiler ere dönüş türülür, unicode veya ansi ve bir ayraç ile ayrılır). Bu bayrak, FileOpen() fonksiyonunda da kullanılır	FileOpen
FILE_END	Dosya sonu işaretini al	FileGetInteger
FILE_EXISTS	Mevcudiyeti kontrol et	FileGetInteger
FILE_IS_ANSI	Dosya, ANSI şeklin	FileGetInteger

Sabit	Açıklama	Usage
	de açılmış (bakınız FILE_ANSI)	
FILE_IS_BINARY	Dosya, ikili dosya olarak açılmış (bakınız, FILE_BIN)	FileGetInteger
FILE_IS_COMMON	Dosya, tüm terminaller tarafından paylaşılan bir klasörde açılmış (bakınız FILE_COMMON)	FileGetInteger
FILE_IS_CSV	Dosya, CSV dosyası olarak açılmış (bakınız FILE_CSV)	FileGetInteger

Sabit	Açıklama	Usage
FILE_IS_READABLE	Açılan dosya okunabilir özellik tedir (bakınız FILE_READ)	FileGetInteger
FILE_IS_TEXT	Dosya, metin dosyası olarak açılmış (bakınız FILE_TXT)	FileGetInteger
FILE_IS_WRITABLE	Açılan dosya yazılabilir özellik tedir (bakınız FILE_WRITE)	FileGetInteger
FILE_LINE_END	Satır sonu işaretini al	FileGetInteger
FILE_MODIFY_DATE	En son değiştirilme tarihi	FileGetInteger
FILE_POSITION	İşaretçinin dosya	FileGetInteger

Sabit	Açıklama	Usage
	içindeki konumu	
FILE_READ	Dosya okuma amaçlı açılır. Bu bayrak FileOpen() fonksiyonunda kullanılır. Bir dosya açarken, FILE_WRITE ve/veya FILE_READ bayraklarının belirtilmesi gerekir.	FileOpen
FILE_REWRITE	FileCopy() ve FileMove() fonksiyonları	FileCopy, FileMove

Sabit	Açıklama	Usage
	kullanılarak dosyanın yeniden yazılması olasılığı. Dosya mevcut olmalı veya yazma amaçlı açılmış olmalıdır, aksi durumda dosya açılmaz.	
FILE_SHARE_READ	Birkaç programdan okunma amacıyla paylaşımlı erişim. Bu bayrak, FileOpen() fonksiyonunda	FileOpen

Sabit	Açıklama	Usage
	kullanılır ama dosya açılması sırasında yine de FILE_WRITE ve/veya FILE_READ bayraklarının belirtilmesi gerekir.	
FILE_SHARE_WRITE	Birkaç programdan yazma amaçlı paylaşımlı erişim. Bu bayrak, FileOpen() fonksiyonunda kullanılır ama dosya açılm	FileOpen

Sabit	Açıklama	Usage
	ası sırasında yine de FILE_WRITE ve/veya FILE_READ bayraklarının belirtilmesi gerekir.	
FILE_SIZE	Bayt bazında dosya boyutu	FileGetInteger
FILE_TXT	Basit metin dosyası (csv dosyasına benzer ama ayraçlar yoktur). Bu bayrak, FileOpen() fonksiyonunda	FileOpen

Sabit	Açıklama	Usage
	kullanılır	
FILE_UNICODE	UNICODE tipi dizgiler (iki baytlık semboller). Bu bayrak, FileOpen() fonksiyonunda kullanılır	FileOpen
FILE_WRITE	Dosya, yazma amacıyla açılır. Bu bayrak FileOpen() fonksiyonunda kullanılır. Bir dosya açarken, FILE_WRITE ve/veya	FileOpen

Sabit	Açıklama	Usage
	FILE_READ bayraklarının belirtilmesi gerekir.	
FLT_DIG	float tipi için anlamlı ondalık hanelerin sayısı	Nümerik Tip Sabitleri
FLT_EPSILON	Şu koşulu sağlayan minimal değer: 1.0+DBL_EPSILON != 1.0 (float tipi için)	Nümerik Tip Sabitleri
FLT_MANT_DIG	float tipi için mantis içindeki bit sayısı	Nümerik Tip Sabitleri
FLT_MAX	float tipiyle temsil	Nümerik Tip Sabitleri

Sabit	Açıklama	Usage
	edilebilecek maksimal değer	
FLT_MAX_10_EXP	float tipli maksimal ondalık üs değeri	Nümerik Tip Sabitleri
FLT_MAX_EXP	float tipli maksimal ikili üs değeri	Nümerik Tip Sabitleri
FLT_MIN	float tipiyle temsil edilebilecek minimal pozitif değer	Nümerik Tip Sabitleri
FLT_MIN_10_EXP	float tipli minimal ondalık üs değeri	Nümerik Tip Sabitleri
FLT_MIN_EXP	float tipli minimal ikili üs değeri	Nümerik Tip Sabitleri
FRIDAY	Cuma	SymbolInfoInteger, SymbolInfoSessionQuote,

Sabit	Açıklama	Usage
		SymbolInfoSessionTrade
GANN_DOWN_TREND	Aşağı yönlü trende karşılık gelen çizgi	ObjectSetInteger, ObjectGetInteger
GANN_UP_TREND	Yukarı yönlü trende karşılık gelen çizgi	ObjectSetInteger, ObjectGetInteger
GATORJAW_LINE	Çene çizgisi	Gösterge Çizgileri
GATORLIPS_LINE	Dudak çizgisi	Gösterge Çizgileri
GATORTEETH_LINE	Diş çizgisi	Gösterge Çizgileri
IDABORT	"Sonlandır" (Abort) düğmesine basılmış	MessageBox
IDCANCEL	"İptal Et" (Cancel) düğmesine basılmış	MessageBox
IDCONTINUE	"Devam Et" (Continue) düğm	MessageBox

Sabit	Açıklama	Usage
	esine basılmış	
IDIGNORE	"Gözardı Et" (Ignore) düğmesine basılmış	MessageBox
IDNO	"Hayır" (No) düğmesine basılmış	MessageBox
IDOK	"Tamam" (OK) düğmesine basılmış	MessageBox
IDRETRY	"Yeniden Dene" (Retry) düğmesine basılmış	MessageBox
IDTRYAGAIN	"Tekrar Dene" (Try Again) düğmesine basılmış	MessageBox

Sabit	Açıklama	Usage
IDYES	"Evet" (Yes) düğmesine basılmış	MessageBox
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

Sabit	Açıklama	Usage
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
IND_CHAIKIN	Chaikin Oscillator	IndicatorCreate, IndicatorParameters
IND_CUSTOM	Özel gösterge	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

Sabit	Açıklama	Usage
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
IND_OSMA	OsMA	IndicatorCreate, IndicatorParameters
IND_RSI	Relative Strength Index	IndicatorCreate, IndicatorParameters
IND_RVI	Relative Vigor Index	IndicatorCreate, IndicatorParameters

Sabit	Açıklama	Usage
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	Ara işleml	SetIndexBuffer

Sabit	Açıklama	Usage
	er için kullanılacak yardımcı tamponlar	
INDICATOR_COLOR_INDEX	Renk	SetIndexBuffer
INDICATOR_DATA	Çizilecek veri	SetIndexBuffer
INDICATOR_DIGITS	Gösterge değerleri çiziminin doğruluğu	IndicatorSetInteger
INDICATOR_HEIGHT	Sabitlenmiş gösterge penceresi yüksekliği (ön işlemci komutu #property indicator_height)	IndicatorSetInteger
INDICATOR_LEVELCOLOR	Seviye çizgisinin rengi	IndicatorSetInteger
INDICATOR_LEVELS	Gösterge	IndicatorSetInteger

Sabit	Açıklama	Usage
	pencere sindeki seviyelerin sayısı	
INDICATOR_LEVELSTYLE	Seviye çizgisinin stili	IndicatorSetInteger
INDICATOR_LEVELTEXT	Seviye açıklaması	IndicatorSetString
INDICATOR_LEVELVALUE	Seviye değeri	IndicatorSetDouble
INDICATOR_LEVELWIDTH	Seviye çizgisinin kalınlığı	IndicatorSetInteger
INDICATOR_MAXIMUM	Gösterge penceresinin maksimum değeri	IndicatorSetDouble
INDICATOR_MINIMUM	Gösterge penceresinin minimum değeri	IndicatorSetDouble
INDICATOR_SHORTNAME	Gösterge	IndicatorSetString

Sabit	Açıklama	Usage
	kısa ismi	
INT_MAX	int tipi ile temsil edilebilecek maksimal değer	Nümerik Tip Sabitleri
INT_MIN	int tipi ile temsil edilebilecek minimal değer	Nümerik Tip Sabitleri
INVALID_HANDLE	Hatalı tanıtıcı değer	Diğer Sabitler
IS_DEBUG_MODE	Bir MQ5 programının hata ayıklama modunda çalıştığını gösteren bayrak	Diğer Sabitler
IS_PROFILE_MODE	Bir MQ5 programının profill	Diğer Sabitler

Sabit	Açıklama	Usage
	eme modunda çalıştığını gösteren bayrak	
KIJUNSEN_LINE	Kijun-sen çizgisi	Gösterge Çizgileri
LICENSE_DEMO	Market içinden, paralı bir ürünün deneme versiyonu. Sadece strateji sınayıcıda çalışır	MQLInfoInteger
LICENSE_FREE	Kullanım sınırlaması olmayan özgür versiyon	MQLInfoInteger
LICENSE_FULL	Satın alınan lisanslı bir versiy	MQLInfoInteger

Sabit	Açıklama	Usage
	on en az 5 aktivasyon sağlar. Satıcı sağlanan aktivasyon sayısını artırabilir	
LICENSE_TIME	Sınırlı süre lisanslı versiyon	MQLInfoInteger
LONG_MAX	long tipi ile temsil edilebilecek maksimal değer	Nümerik Tip Sabitleri
LONG_MIN	long tipi ile temsil edilebilecek minimal değer	Nümerik Tip Sabitleri
LOWER_BAND	Alt limit	Gösterge Çizgileri
LOWER_HISTOGRAM	Alt histogram	Gösterge Çizgileri

Sabit	Açıklama	Usage
LOWER_LINE	Alt çizgi	Gösterge Çizgileri
M_1_PI	1/pi	Matematiksel Sabitler
M_2_PI	2/pi	Matematiksel Sabitler
M_2_SQRTPI	2/sqrt(pi)	Matematiksel Sabitler
M_E	e	Matematiksel Sabitler
M_LN10	ln(10)	Matematiksel Sabitler
M_LN2	ln(2)	Matematiksel Sabitler
M_LOG10E	log10(e)	Matematiksel Sabitler
M_LOG2E	log2(e)	Matematiksel Sabitler
M_PI	pi	Matematiksel Sabitler
M_PI_2	pi/2	Matematiksel Sabitler
M_PI_4	pi/4	Matematiksel Sabitler
M_SQRT1_2	1/sqrt(2)	Matematiksel Sabitler
M_SQRT2	sqrt(2)	Matematiksel Sabitler
MAIN_LINE	Ana çizgi	Gösterge Çizgileri
MB_ABORTRETRYIGNORE	Üç düğme içeren mesaj pencere si: Abort, Retry ve Ignore	MessageBox
MB_CANCELTRYCONTINUE	Üç düğme içeren mesaj	MessageBox

Sabit	Açıklama	Usage
	pencere si: Cancel, Try Again, Continue	
MB_DEFBUTTON1	İlk düğme MB_DEFBUTTON1 - eğer MB_DEFBUTTON2, MB_DEFBUTTON3 veya MB_DEFBUTTON4 belirtilmemişse öntanımlıdır	MessageBox
MB_DEFBUTTON2	İkinci düğme öntanımlıdır	MessageBox
MB_DEFBUTTON3	Üçüncü düğme öntanımlıdır	MessageBox
MB_DEFBUTTON4	Dördüncü düğme ön	MessageBox

Sabit	Açıklama	Usage
	tanımlıdır	
MB_ICONEXCLAMATION, MB_ICONWARNING	Ünlem işareti ikonu	MessageBox
MB_ICONINFORMATION, MB_ICONASTERISK	Daire içine alınmış i işareti	MessageBox
MB_ICONQUESTION	Soru işareti ikonu	MessageBox
MB_ICONSTOP, MB_ICONERROR, MB_ICONHAND	STOP işareti ikonu	MessageBox
MB_OK	Sadece bir düğme içeren mesaj pencere si: OK. Default	MessageBox
MB_OKCANCEL	İki düğme içeren mesaj pencere si: OK ve Cancel	MessageBox
MB_RETRYCANCEL	İki düğm	MessageBox

Sabit	Açıklama	Usage
	e içeren mesaj pencere resi: Retry ve Cancel	
MB_YESNO	İki düğme içeren mesaj pencere resi: Yes ve No	MessageBox
MB_YESNOCANCEL	Üç düğme içeren mesaj pencere resi: Yes, No ve Cancel	MessageBox
MINUSDI_LINE	Çizgi -DI	Gösterge Çizgileri
MODE_EMA	Üssel ortalama	Düzleştirme Yöntemleri
MODE_LWMA	Doğrusal-ağırlıklı ortalama	Düzleştirme Yöntemleri
MODE_SMA	Basit ortalama	Düzleştirme Yöntemleri

Sabit	Açıklama	Usage
MODE_SMMA	Düzleştirilmiş ortalama	Düzleştirme Yöntemleri
MONDAY	Pazartesi	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
MQL_DEBUG	Hata ayıklama modunu belirten bayrak	MQLInfoInteger
MQL_DLLS_ALLOWED	Verilen çalıştırılmış program için DLL kullanım izni	MQLInfoInteger
MQL_FRAME_MODE	Uzman Danışmanın optimizasyon sonuç çerçeveleri nin toplanması modunda çalıştığını	MQLInfoInteger

Sabit	Açıklama	Usage
	ifade eden bayrak	
MQL_LICENSE_TYPE	EX5 modülünün lisans tipi. Lisans, Mql5InfoInteger(MQL_LICENSE_TYPE) kullanımıyla bir isteğin yapıldığı EX5 modülünü ifade eder.	MQLInfoInteger
MQL_MEMORY_LIMIT	Terminalde aynı anda açık olabilecek maksimum çizelge sayısı	MQLInfoInteger
MQL_MEMORY_USED	Temsilci tarafından	MQLInfoInteger

Sabit	Açıklama	Usage
	kullanılan bellek miktarı, MB	
MQL_OPTIMIZATION	Optimizasyon sürecini belirten bayrak	MQLInfoInteger
MQL_PROFILER	Programın kod profilleme modunda işletildiğini belirten bayrak	MQLInfoInteger
MQL_PROGRAM_NAME	Çalıştırılan MQL5 programının ismi	MQLInfoString
MQL_PROGRAM_PATH	Verilen çalıştırılmış program için dosya yolu	MQLInfoString
MQL_PROGRAM_TYPE	MQL5 progr	MQLInfoInteger

Sabit	Açıklama	Usage
	amının tipi	
MQL_SIGNALS_ALLOWED	The permission to modify the Signals for the given executed program	MQLInfoInteger
MQL_TESTER	Test sürecini belirten bayrak	MQLInfoInteger
MQL_TRADE_ALLOWED	Verilen çalıştırılmış program için alım-satım izni	MQLInfoInteger
MQL_VISUAL_MODE	Görsel test sürecini belirleyen bayrak	MQLInfoInteger
NULL	Tüm tipler için	Diğer Sabitler

Sabit	Açıklama	Usage
	sıfırdır	
OBJ_ALL_PERIODS	Nesne tüm zaman dilimlerinde çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_ARROW	Ok	Nesne Tipleri
OBJ_ARROW_BUY	Alış Sinyali	Nesne Tipleri
OBJ_ARROW_CHECK	Kontrol İşareti	Nesne Tipleri
OBJ_ARROW_DOWN	Aşağı Ok	Nesne Tipleri
OBJ_ARROW_LEFT_PRICE	Sol Fiyat Etiketi	Nesne Tipleri
OBJ_ARROW_RIGHT_PRICE	Sağ Fiyat Etiketi	Nesne Tipleri
OBJ_ARROW_SELL	Satış Sinyali	Nesne Tipleri
OBJ_ARROW_STOP	Dur (Stop) İşareti	Nesne Tipleri
OBJ_ARROW_THUMB_DOWN	Kötü (Başparmak Aşağı)	Nesne Tipleri
OBJ_ARROW_THUMB_UP	İyi (Başparmak	Nesne Tipleri

Sabit	Açıklama	Usage
	Yukarı)	
OBJ_ARROW_UP	Yukarı Ok	Nesne Tipleri
OBJ_ARROWED_LINE	Oklu çizgi	Nesne Tipleri
OBJ_BITMAP	Bitmap Etiketi	Nesne Tipleri
OBJ_BITMAP_LABEL	Bitmap Etiketi	Nesne Tipleri
OBJ_BUTTON	Düğme	Nesne Tipleri
OBJ_CHANNEL	Eşit-Aralıklı Kanal	Nesne Tipleri
OBJ_CHART	Çizelge	Nesne Tipleri
OBJ_CYCLES	Döngü Çizgileri	Nesne Tipleri
OBJ_EDIT	Düzenle	Nesne Tipleri
OBJ_ELLIOTWAVE3	Elliott Düzeltme Dalgası	Nesne Tipleri
OBJ_ELLIOTWAVE5	Elliott Dürtü Dalgası	Nesne Tipleri
OBJ_ELLIPSE	Elips	Nesne Tipleri
OBJ_EVENT	Ekonomik takvi	Nesne Tipleri

Sabit	Açıklama	Usage
	mdeki bir olaya karşılık gelen "Olay" nesnesi	
OBJ_EXPANSION	Fibonacci Açılımı	Nesne Tipleri
OBJ_FIBO	Fibonacci Düzeltme Seviyeleri	Nesne Tipleri
OBJ_FIBOARC	Fibonacci Yayları	Nesne Tipleri
OBJ_FIBOCHANNEL	Fibonacci Kanalı	Nesne Tipleri
OBJ_FIBOFAN	Fibonacci Yelpazesi	Nesne Tipleri
OBJ_FIBOTIMES	Fibonacci Zaman Dilimleri	Nesne Tipleri
OBJ_GANNFAN	Gann Yelpazesi	Nesne Tipleri
OBJ_GANNGRID	Gann Izgarası	Nesne Tipleri

Sabit	Açıklama	Usage
OBJ_GANNLINE	Gann Çizgisi	Nesne Tipleri
OBJ_HLINE	Yatay Çizgi	Nesne Tipleri
OBJ_LABEL	Etiket	Nesne Tipleri
OBJ_NO_PERIODS	Nesne tüm zaman dilimlerinde çizilmez	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_D1	Nesne günlük çizelgelerde çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H1	Nesne 1-saatlik çizelgede çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H12	Nesne 12-saatlik çizelgede çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H2	Nesne 2-saatlik çizelgede çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H3	Nesne 3-	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	saatlik çizelgede çizilir	
OBJ_PERIOD_H4	Nesne 4-saatlik çizelgede çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H6	Nesne 6-saatlik çizelgede çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_H8	Nesne 8-saatlik çizelgede çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M1	Nesne 1-dakikalık çizelgede çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M10	Nesne 10-dakikalık çizelgede çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M12	Nesne 12-dakikalık	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	çizelgede çizilir	
OBJ_PERIOD_M15	Nesne 15-dakikalık çizelgede çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M2	Nesne 2-dakikalık çizelgede çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M20	Nesne 20-dakikalık çizelgede çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M3	Nesne 3-dakikalık çizelgede çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M30	Nesne 30-dakikalık çizelgede çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M4	Nesne 4-dakikalık çizelg	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	ede çizilir	
OBJ_PERIOD_M5	Nesne 5-dakikalık çizelgede çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_M6	Nesne 6-dakikalık çizelgede çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_MN1	Nesne haftalık çizelgelerde çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PERIOD_W1	Nesne haftalık çizelgelerde çizilir	ObjectSetInteger, ObjectGetInteger
OBJ_PITCHFORK	Andrews Çatalı (Üçlü Çizgi)	Nesne Tipleri
OBJ_RECTANGLE	Dikdörtgen	Nesne Tipleri
OBJ_RECTANGLE_LABEL	Özel grafiksel arayüzler oluşturmak ve	Nesne Tipleri

Sabit	Açıklama	Usage
	tasarlamak için "Dikdörtgen Etiket" nesnesi.	
OBJ_REGRESSION	Doğrusal Regresyon Kanalı	Nesne Tipleri
OBJ_STDDEVCHANNEL	Standart Sapma Kanalı	Nesne Tipleri
OBJ_TEXT	Metin	Nesne Tipleri
OBJ_TREND	Trend Çizgisi	Nesne Tipleri
OBJ_TRENDBYANGLE	Açılı Trend Çizgisi	Nesne Tipleri
OBJ_TRIANGLE	Üçgen	Nesne Tipleri
OBJ_VLINE	Dikey Çizgi	Nesne Tipleri
OBJPROP_ALIGN	Edit nesnesinde (OBJ_EDIT) yatay metin hizalama	ObjectSetInteger, ObjectGetInteger
OBJPROP_ANCHOR	Bir grafik	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	sel nesnenin tuttur ma noktası konumu	
OBJPROP_ANGLE	Açı. Bir progr am tarafı ndan açı belirti lmede n oluştu rulmu ş nesnel er için , boş değer e eşittir EMPT Y_VAL UE	ObjectSetDouble, ObjectGetDouble
OBJPROP_ARROWCODE	Ok nesne si için, ok kodu	ObjectSetInteger, ObjectGetInteger
OBJPROP_BACK	Arkapl andak i nesne	ObjectSetInteger, ObjectGetInteger
OBJPROP_BGCOLOR	OBJ_E DIT, OBJ_B UTTO	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	N ve OBJ_RECTANGLE_LABEL için arkaplan rengi	
OBJPROP_BMPFILE	Bitmap etiketi için BMP-dosyasının ismi. Bakınız Kaynaklar	ObjectSetString, ObjectGetString
OBJPROP_BORDER_COLOR	OBJ_EDIT ve OBJ_BUTTON nesneleri için kenar çizgisi rengi	ObjectSetInteger, ObjectGetInteger
OBJPROP_BORDER_TYPE	"Dikdörtgen etiket" nesnesi için kenar tipi	ObjectSetInteger, ObjectGetInteger
OBJPROP_CHART_ID	"Çizelge" nesne	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	sinin (OBJ_CHART) tanımlayıcısı. Bu, çizelge işlemlerinde tarif edilen fonksiyonlar kullanılarak, nesnenin özellikleri ile normal bir çizelgeymiş gibi çalışılmasını sağlarama bu duruma bazı istisnalar vardır.	
OBJPROP_CHART_SCALE	Çizelge nesnesi için ölçek	ObjectSetInteger, ObjectGetInteger
OBJPROP_COLOR	Renk	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
OBJPROP_CORNER	Grafiksel nesnenin bağlanabileceği çizelge köşesi	ObjectSetInteger, ObjectGetInteger
OBJPROP_CREATETIME	Nesnenin oluşturulma zamanı	ObjectSetInteger, ObjectGetInteger
OBJPROP_DATE_SCALE	Çizelge nesnesi için zaman ölçeğinin görüntülenmesi	ObjectSetInteger, ObjectGetInteger
OBJPROP_DEGREE	Elliott Dalga İşaretinin seviyesi	ObjectSetInteger, ObjectGetInteger
OBJPROP_DEVIATION	Standart Sapma Kanalı için sapma değeri	ObjectSetDouble, ObjectGetDouble
OBJPROP_DIRECTION	Gann nesne	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	sinin trendi	
OBJPROP_DRAWLINES	Elliott Dalgasının işaretlenecek çizgileri görüntülemesi	ObjectSetInteger, ObjectGetInteger
OBJPROP_ELLIPSE	Fibonacci Yay nesnesi (OBJ_FIBOARC) için bütün bir elipsi gösterir	ObjectSetInteger, ObjectGetInteger
OBJPROP_FILL	Nesneyi renkle doldur (OBJ_RECTANGLE, OBJ_TRIANGLE, OBJ_ELLIPSE, OBJ_CHANNEL, OBJ_STDDE	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	VCHANNEL ve OBJ_REGRESSION nesneleri için)	
OBJPROP_FONT	Yazı tipi	ObjectSetString, ObjectGetString
OBJPROP_FONTSIZE	Yazı tipi boyutu	ObjectSetInteger, ObjectGetInteger
OBJPROP_HIDDEN	Nesne listesi içindeki bir grafiksel nesnenin ismini n, terminalin "Grafikler" - "Nesneler" - "nesne listesi" menüsünde gösterimini yasaklar. true değeri nesnenin listed	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	e gizlenmesini sağlar. Varsayılan olarak, takvim olayları, alım-satım geçmişi olayları ve MQL5 programları ile oluşturulmuş olaylar için 'true' değeri ayarlanmıştır. Bunlar gibi grafiksel nesneleri görüntülemek ve özelliklerine erişebilmek için,	

Sabit	Açıklama	Usage
	"nesne listesi" pencere sindeki "Tümü" düğmesine basın.	
OBJPROP_LEVELCOLOR	Seviye çizgisinin rengi	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELS	Seviyelerin sayısı	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELSTYLE	Seviye çizgisinin stili stili	ObjectSetInteger, ObjectGetInteger
OBJPROP_LEVELTEXT	Seviye açıklaması	ObjectSetString, ObjectGetString
OBJPROP_LEVELVALUE	Seviye değeri	ObjectSetDouble, ObjectGetDouble
OBJPROP_LEVELWIDTH	Seviye çizgisinin kalınlığı	ObjectSetInteger, ObjectGetInteger
OBJPROP_NAME	Nesne ismi	ObjectSetString, ObjectGetString

Sabit	Açıklama	Usage
OBJPROP_PERIOD	"Çizelge" nesnesi için zaman aralığı	ObjectSetInteger, ObjectGetInteger
OBJPROP_PRICE	Fiyat koordinatı	ObjectSetDouble, ObjectGetDouble
OBJPROP_PRICE_SCALE	Çizelge nesnesi için fiyat ölçeğinin görüntülenmesi	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY	Tüm çizelge pencerelerinin içinden geçen dikey çizgi	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY_LEFT	Işın sola gider	ObjectSetInteger, ObjectGetInteger
OBJPROP_RAY_RIGHT	Işın sağa gider	ObjectSetInteger, ObjectGetInteger
OBJPROP_READONLY	Edit nesnesindeki metnin düzenl	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	enme olanağı	
OBJPROP_SCALE	Ölçek (Gann nesnelerinin ve Fibonacci Yaylarının özellikleri)	ObjectSetDouble, ObjectGetDouble
OBJPROP_SELECTABLE	Nesnenin mevcudiyeti	ObjectSetInteger, ObjectGetInteger
OBJPROP_SELECTED	Nesle seçildi	ObjectSetInteger, ObjectGetInteger
OBJPROP_STATE	Düğme durumu (basılı / serbest)	ObjectSetInteger, ObjectGetInteger
OBJPROP_STYLE	Stil	ObjectSetInteger, ObjectGetInteger
OBJPROP_SYMBOL	Çizelge nesnesinin sembolü	ObjectSetString, ObjectGetString
OBJPROP_TEXT	Nesne açıklaması (nesne içindeki	ObjectSetString, ObjectGetString

Sabit	Açıklama	Usage
	metin)	
OBJPROP_TIME	Zaman koordinatı	ObjectSetInteger, ObjectGetInteger
OBJPROP_TIMEFRAMES	Bir nesnenin farklı zaman dilimlerindeki görünürlüğü	ObjectSetInteger, ObjectGetInteger
OBJPROP_TOOLTIP	Araç ipucu metni. Eğer özellik ayarlanmamışsa, o zaman araç ipucu metni, terminal tarafından otomatik olarak ayarlanır. Bir araç ipucu "\n" (satır başı	ObjectSetString, ObjectGetString

Sabit	Açıklama	Usage
	karakteri) değerinin ayarlanmasıyla devre dışı bırakılabilir	
OBJPROP_TYPE	Nesne tipi	ObjectSetInteger, ObjectGetInteger
OBJPROP_WIDTH	Çizgi kalınlığı	ObjectSetInteger, ObjectGetInteger
OBJPROP_XDISTANCE	Bağlama köşesinden itibaren, X ekseni üzerinde piksel bazında uzaklık (see note)	ObjectSetInteger, ObjectGetInteger
OBJPROP_XOFFSET	"Bitmap Label" ve "Bitmap" (OBJ_BITMAP_LABEL ve OBJ_BITMAP)	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	grafiksel nesnelerindeki görünür dikdörtgen alanının sol üst köşesinin X koordinatları. Bu değer, orjinal görüntünün üst sol köşesine göre piksel bazında ayarlanır.	
OBJPROP_XSIZE	X ekseni üzerinde, piksel bazında nesne genişliği. OBJ_LABEL (read	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT ve OBJ_RECTANGLE_LABEL nesneleri için belirlenmiştir.	
OBJPROP_YDISTANCE	Bağlama köşesinden itibaren, Y ekseni üzerinde piksel bazında uzaklık (see note)	ObjectSetInteger, ObjectGetInteger
OBJPROP_YOFFSET	"Bitmap	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	Label" ve "Bitmap" (OBJ_BITMAP_LABEL ve OBJ_BITMAP) grafiksel nesnelerindeki görünür dikdörtgen alanının sol üst köşesinin Y koordinatları. Bu değer, orjinal görüntünün üst sol köşesine göre piksel bazında ayarlanır.	
OBJPROP_YSIZE	Y ekseni	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	üzerinde, piksel bazında nesne yüksekliği. OBJ_LABEL (read only), OBJ_BUTTON, OBJ_CHART, OBJ_BITMAP, OBJ_BITMAP_LABEL, OBJ_EDIT ve OBJ_RECTANGLE_LABEL nesneleri için belirlenmiştir.	
OBJPROP_ZORDER	Çizelge üzerine tıklama olayının	ObjectSetInteger, ObjectGetInteger

Sabit	Açıklama	Usage
	(CHARTEVENT_CLICK) için, bir grafiksel nesnenin önceliği. Bu değer, nesne oluşturulurken ön tanımlı olarak sıfıra ayarlanır; gerektiğinde öncelik değiştirilebilir. Bir nesneyi diğerinin üzerine uygularken, CHARTEVENT_CLICK olayın, bunların	

Sabit	Açıklama	Usage
	sadece en yüksek önceliklikli olanı alacaktır.	
ORDER_COMMENT	Emir yorumu	OrderGetString, HistoryOrderGetString
ORDER_FILLING_FOK	Bu emir türü, emrin sadece belirtilen hacimle karşılanabileceği anlamına gelir. Eğer piyasada yeterli miktarda finansal enstrüman bulunmuyorsa, emir yürürlüğe konulmaz.	OrderGetInteger, HistoryOrderGetInteger

Sabit	Açıklama	Usage
	İstenen hacim değeri, o an piyasada mevcut olan birkaç teklif ile karşılanabilir.	
ORDER_FILLING_IOC	Bu mod kullanıcının, emirde belirtilen sınırlar içerisinde piyasada bulunan maksimum hacim değeri ile işlem yapmaya razı olmasını anlamına gelir. Bu	OrderGetInteger, HistoryOrderGetInteger

Sabit	Açıklama	Usage
	durumda emirde belirtilen hacmin tamamı karşılanamayabilir, mevcut olan hacim değeri kadar karşılama yapılır, kalanı ise iptal edilir.	
ORDER_FILLING_RETURN	Bu tür, sadece piyasa emirleri (ORDER_TYPE_BUY ve ORDER_TYPE_SELL), limit ve stop limit	OrderGetInteger, HistoryOrderGetInteger

Sabit	Açıklama	Usage
	emirleri (ORDER_TYPE_BUY_LIMIT, ORDER_TYPE_SELL_LIMIT, ORDER_TYPE_BUY_STOP_LIMIT ve ORDER_TYPE_SELL_STOP_LIMIT) için ve sadece Piyasa veya Borsa işlemleri için kullanılır. Kısmi karşılanma durumunda, piyasa veya limit emrin in kalan kısmı	

Sabit	Açıklama	Usage
	iptal edilmez, bununyerinedahasonraişlenir. ORDER_TYPE_BUY_STOP_LIMIT ve ORDER_TYPE_SELL_STOP_LIMIT emirlerininaktivasyonuiçin, karşılık gelen ORDER_FILLING_RETURN işlem tipli bir ORDER_TYPE_BUY_LIMIT/ORDER_TYPE_SELL_LIMIT limit emri	

Sabit	Açıklama	Usage
	oluşturulur.	
ORDER_MAGIC	Emri veren Uzman Danışmanın tanımlayıcısı (her bir Uzman Danışmanın kendine has, benzersiz numarasını girmesi için tasarlanmıştır)	OrderGetInteger, HistoryOrderGetInteger
ORDER_POSITION_ID	Pozisyon tanımlayıcısı, gerçekleştiği anda bir emir için ayarlanır. Gerçekleştirilen her	OrderGetInteger, HistoryOrderGetInteger

Sabit	Açıklama	Usage
	emir, mevcut bir pozisyonu açan veya değiştiren bir işlem ile sonuçlanır. Pozisyon tanımlayıcısı çalıştırılan emir için, işlem anında ayarlanır.	
ORDER_PRICE_CURRENT	Emir verilen sembolün mevcut fiyatı	OrderGetDouble, HistoryOrderGetDouble
ORDER_PRICE_OPEN	Emirde belirtilen fiyat	OrderGetDouble, HistoryOrderGetDouble
ORDER_PRICE_STOPLIMIT	StopLimit emri için Limitl	OrderGetDouble, HistoryOrderGetDouble

Sabit	Açıklama	Usage
	i emir fiyatı	
ORDER_SL	Zarar Durdur değeri	OrderGetDouble, HistoryOrderGetDouble
ORDER_STATE	Emir durumu	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_CANCELED	Emir müşteri tarafından iptal edildi	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_EXPIRED	Emir, zaman-aşımına uğradı	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_FILLED	Emir bütünüyle işlendi	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_PARTIAL	Emir kısmen işlendi	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_PLACED	Emir kabul edildi	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REJECTED	Emir reddedildi	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_ADD	Emir işleniyor (alım-	OrderGetInteger, HistoryOrderGetInteger

Sabit	Açıklama	Usage
	satım sistemine kaydediliyor)	
ORDER_STATE_REQUEST_CANCEL	Emir siliniyor (alım-satım sisteminden siliniyor)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_REQUEST_MODIFY	Emir şekillendiriliyor (parametreleri değiştiriliyor)	OrderGetInteger, HistoryOrderGetInteger
ORDER_STATE_STARTED	Emir kontrol edilmiş ama henüz aracı kurum tarafından kabul edilmemiş	OrderGetInteger, HistoryOrderGetInteger
ORDER_SYMBOL	Emrin verildiği sembol	OrderGetString, HistoryOrderGetString

Sabit	Açıklama	Usage
ORDER_TIME_DAY	Günlük emir	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_DONE	Emrin gerçekleşme veya iptal edilme zamanı	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_DONE_MSC	01.01.1970 beri geçen milisaniyeler cinsinden emrin gerçekleşme/iptal edilme zamanı	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_EXPIRATION	Emir zaman-aşımı tipi	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_GTC	İptale kadar geçerli emir	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SETUP	Emir başlangıç zamanı	OrderGetInteger, HistoryOrderGetInteger

Sabit	Açıklama	Usage
ORDER_TIME_SETUP_MSC	01.01.1970 beri geçen milisaniyeler cinsinden emrin girilme zamanı	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SPECIFIED	Tarihli emir	OrderGetInteger, HistoryOrderGetInteger
ORDER_TIME_SPECIFIED_DAY	Emir, belirtilen günün 23:59:59 saatine kadar aktif olacaktır. Eğer belirtilen zaman, alım-satım seansının dışında kalıyorsa; emir, en yakın alım-satım	OrderGetInteger, HistoryOrderGetInteger

Sabit	Açıklama	Usage
	anında zaman aşımına uğrayacaktır.	
ORDER_TP	Kar Al değeri	OrderGetDouble, HistoryOrderGetDouble
ORDER_TYPE	Emir tipi	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY	Piyasa Alış emri	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_LIMIT	Buy Limit bekleyen emri	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP	Buy Stop bekleyen emri	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_BUY_STOP_LIMIT	Emir fiyatına ulaşıldığında, bir Buy Limit bekleyen emri StopLimit fiyatına yerleştirilir	OrderGetInteger, HistoryOrderGetInteger

Sabit	Açıklama	Usage
ORDER_TYPE_FILLING	Emrin karşılanma tipi	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL	Piyasa Satış emri	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_LIMIT	Sell Limit bekleyen emri	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_STOP	Sell Stop bekleyen emri	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_SELL_STOP_LIMIT	Emir fiyatına ulaşıldığında, bir Sell Limit bekleyen emri StopLimit fiyatına yerleştirilir	OrderGetInteger, HistoryOrderGetInteger
ORDER_TYPE_TIME	Emrin yaşam süresi	OrderGetInteger, HistoryOrderGetInteger
ORDER_VOLUME_CURRENT	Emrin mevcut hacmi	OrderGetDouble, HistoryOrderGetDouble
ORDER_VOLUME_INITIAL	Emrin başlan	OrderGetDouble, HistoryOrderGetDouble

Sabit	Açıklama	Usage
	gıç hacmi	
PERIOD_CURRENT	Mevcut zaman aralığı	Çizelge Zaman-Aralıkları
PERIOD_D1	1 gün	Çizelge Zaman-Aralıkları
PERIOD_H1	1 saat	Çizelge Zaman-Aralıkları
PERIOD_H12	12 saat	Çizelge Zaman-Aralıkları
PERIOD_H2	2 saat	Çizelge Zaman-Aralıkları
PERIOD_H3	3 saat	Çizelge Zaman-Aralıkları
PERIOD_H4	4 saat	Çizelge Zaman-Aralıkları
PERIOD_H6	6 saat	Çizelge Zaman-Aralıkları
PERIOD_H8	8 saat	Çizelge Zaman-Aralıkları
PERIOD_M1	1 dakika	Çizelge Zaman-Aralıkları
PERIOD_M10	10 dakika	Çizelge Zaman-Aralıkları
PERIOD_M12	12 dakika	Çizelge Zaman-Aralıkları
PERIOD_M15	15 dakika	Çizelge Zaman-Aralıkları
PERIOD_M2	2 dakika	Çizelge Zaman-Aralıkları
PERIOD_M20	20 dakika	Çizelge Zaman-Aralıkları
PERIOD_M3	3 dakika	Çizelge Zaman-Aralıkları

Sabit	Açıklama	Usage
PERIOD_M30	30 dakika	Çizelge Zaman-Aralıkları
PERIOD_M4	4 dakika	Çizelge Zaman-Aralıkları
PERIOD_M5	5 dakika	Çizelge Zaman-Aralıkları
PERIOD_M6	6 dakika	Çizelge Zaman-Aralıkları
PERIOD_MN1	1 ay	Çizelge Zaman-Aralıkları
PERIOD_W1	1 hafta	Çizelge Zaman-Aralıkları
PLOT_ARROW	DRAW_ARROW stili için ok kodu	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_ARROW_SHIFT	DRAW_ARROW stilinde okların dikey olarak kaydırılması	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_COLOR_INDEXES	Renk sayısı	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_DRAW_BEGIN	Çizim yapılmamış çubuk sayısı ve	PlotIndexSetInteger, PlotIndexGetInteger

Sabit	Açıklama	Usage
	Veri Pencesindeki değerler	
PLOT_DRAW_TYPE	Grafiksel çizim tipi	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_EMPTY_VALUE	Çizim yapılmayan bir grafik için boş değer	PlotIndexSetDouble
PLOT_LABEL	Veri Pencesinde gösterilecek grafiksel serinin ismi. Birkaç gösterge tamponunun kullanımını gerektiren karmaşık grafik stilleri ile çalışırken, her	PlotIndexSetString

Sabit	Açıklama	Usage
	bir tamponun ismi ";" ayracı kullanılarak belirtilebilir. Örnek kod şurada gösterilmektedir DRAW_CANDLES	
PLOT_LINE_COLOR	Çizim rengini içeren bir tamponun indisi	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_LINE_STYLE	Çizgi çizim stili	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_LINE_WIDTH	Çizgi çizim kalınlığı	PlotIndexSetInteger, PlotIndexGetInteger
PLOT_SHIFT	Gösterge grafiğinin, zaman ekseninde çubuk	PlotIndexSetInteger, PlotIndexGetInteger

Sabit	Açıklama	Usage
	bazında kaydırılması	
PLOT_SHOW_DATA	Çizim değerlerinin Veri Penceresinde görüntülenen işareti	PlotIndexSetInteger, PlotIndexGetInteger
PLUSDI_LINE	Çizgi +DI	Gösterge Çizgileri
POINTER_AUTOMATIC	Otomatik olarak (new() kullanılmadan) oluşturulmuş herhangi bir nesnenin işaretçisi	CheckPointer
POINTER_DYNAMIC	new() operatörü ile oluşturulmuş nesnenin	CheckPointer

Sabit	Açıklama	Usage
	işaretçisi	
POINTER_INVALID	Hatalı işaretçi	CheckPointer
POSITION_COMMENT	Pozisyon yorumu	PositionGetString
POSITION_COMMISSION	Komisyon	PositionGetDouble
POSITION_IDENTIFIER	Pozisyon tanımlayıcısı (kimliği), yeni açılan her pozisyona atanan ve pozisyonun ömrü boyunca değişmeyen, benzersiz bir sayıdır Pozisyonun devri, tanımlayıcıyı değiştirmez.	PositionGetInteger

Sabit	Açıklama	Usage
POSITION_MAGIC	Pozisyon magic number (bakınız ORDER_MAGIC)	PositionGetInteger
POSITION_PRICE_CURRENT	Pozisyon sembolünün mevcut fiyatı	PositionGetDouble
POSITION_PRICE_OPEN	Pozisyon açılış fiyatı	PositionGetDouble
POSITION_PROFIT	Mevcut kar	PositionGetDouble
POSITION_SL	Açık Pozisyon için Zarar Durdur seviyesi	PositionGetDouble
POSITION_SWAP	Birikmiş swap	PositionGetDouble
POSITION_SYMBOL	Pozisyon sembolü	PositionGetString
POSITION_TIME	Pozisyon açılış	PositionGetInteger

Sabit	Açıklama	Usage
	zamanı	
POSITION_TIME_MSC	01.01.1970 tarihinden beri geçen milisaniyeler şeklinde, pozisyon açılış zamanı	PositionGetInteger
POSITION_TIME_UPDATE	01.01.1970 tarihinden beri geçen saniyeler şeklinde, pozisyon değişim zamanı	PositionGetInteger
POSITION_TIME_UPDATE_MSC	01.01.1970 tarihinden beri geçen milisaniyeler şeklinde, pozisy	PositionGetInteger

Sabit	Açıklama	Usage
	on değişim zamanı	
POSITION_TP	Açık pozisyon için Kar Al seviyesi	PositionGetDouble
POSITION_TYPE	Pozisyon tipi	PositionGetInteger
POSITION_TYPE_BUY	Alış işlemi	PositionGetInteger
POSITION_TYPE_SELL	Satış	PositionGetInteger
POSITION_VOLUME	Pozisyon Hacmi	PositionGetDouble
PRICE_CLOSE	Kapanış fiyatı	Fiyat Sabitleri
PRICE_HIGH	Periyot içi maksimum fiyat	Fiyat Sabitleri
PRICE_LOW	Periyot içi maksimum fiyat	Fiyat Sabitleri
PRICE_MEDIAN	Medyan fiyat, (high + low)/2	Fiyat Sabitleri

Sabit	Açıklama	Usage
PRICE_OPEN	Açılış fiyatı	Fiyat Sabitleri
PRICE_TYPICAL	Tipik fiyat, (high + low + close) /3	Fiyat Sabitleri
PRICE_WEIGHTED	Ağırlıklı ortalama fiyat, (high + low + close + close) /4	Fiyat Sabitleri
PROGRAM_EXPERT	Uzman	MQLInfoInteger
PROGRAM_INDICATOR	Gösterge	MQLInfoInteger
PROGRAM_SCRIPT	Script	MQLInfoInteger
REASON_ACCOUNT	Yeni bir hesap aktive edilmiş veya hesap ayarlarındaki bir değişiklik nedeniyle alım-satım sunucusuna	UninitializeReason, OnDeinit

Sabit	Açıklama	Usage
	yeniden bağlanılmış	
REASON_CHARTCHANGE	Sembol veya çizelge periyodu değişmiş	UninitializeReason, OnDeinit
REASON_CHARTCLOSE	Çizelge kapatılmış	UninitializeReason, OnDeinit
REASON_CLOSE	Terminal kapatılmış	UninitializeReason, OnDeinit
REASON_INITFAILED	Bu değer, OnInit() işleyicisinin sıfır olmayan bir değere dönüş yaptığı anlamına gelir	UninitializeReason, OnDeinit
REASON_PARAMETERS	Giriş parametreleri kullanıcı	UninitializeReason, OnDeinit

Sabit	Açıklama	Usage
	tarafından değiştirilmiş	
REASON_PROGRAM	Uzman Danışmanın kendi işlemini ExpertRemove() fonksiyon çağrısıyla sonlandırmış	UninitializeReason, OnDeinit
REASON_RECOMPILE	Program yeniden derlenmiş	UninitializeReason, OnDeinit
REASON_REMOVE	Program çizelgeden silinmiş	UninitializeReason, OnDeinit
REASON_TEMPLATE	Yeni bir şablon uygulanmış	UninitializeReason, OnDeinit
SATURDAY	Cumartesi	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
SEEK_CUR	Dosya işaretçisinin	FileSeek

Sabit	Açıklama	Usage
	mevcut konumu	
SEEK_END	Dosya sonu	FileSeek
SEEK_SET	Dosya başlangıcı	FileSeek
SENKOUSPANA_LINE	Senkou Span A çizgisi	Gösterge Çizgileri
SENKOUSPANB_LINE	Senkou Span B çizgisi	Gösterge Çizgileri
SERIES_BARS_COUNT	Mevcut anda, belirli bir sembol-periyot için çubuk sayısı	SeriesInfoInteger
SERIES_FIRSTDATE	Mevcut anda, belirli bir sembol oldeki-periyottaki ilk tarih	SeriesInfoInteger

Sabit	Açıklama	Usage
SERIES_LASTBAR_DATE	Sembol deki - periyottaki son çubuğun açılış zamanı	SeriesInfoInteger
SERIES_SERVER_FIRSTDATE	Zaman aralığından bağımsız olarak, sembolün sunucuda bulunan geçmişindeki ilk tarih	SeriesInfoInteger
SERIES_SYNCHRONIZED	Mevcut an için belirli bir sembol deki/periyottaki veri için senkronizasyon bayrağı	SeriesInfoInteger

Sabit	Açıklama	Usage
SERIES_TERMINAL_FIRSTDATE	Zaman aralığından bağımsız olarak, sembolün terminalde bulunan geçmişindeki ilk tarih	SeriesInfoInteger
SHORT_MAX	short tipi ile temsil edilebilecek maksimal değer	Nümerik Tip Sabitleri
SHORT_MIN	uchar tipi ile temsil edilebilecek minimal değer	Nümerik Tip Sabitleri
SIGNAL_BASE_AUTHOR_LOGIN	Author login	SignalBaseGetString
SIGNAL_BASE_BALANCE	Account balance	SignalBaseGetDouble
SIGNAL_BASE_BROKER	Broker name	SignalBaseGetString

Sabit	Açıklama	Usage
	(company)	
SIGNAL_BASE_BROKER_SERVER	Broker server	SignalBaseGetString
SIGNAL_BASE_CURRENCY	Signal base currency	SignalBaseGetString
SIGNAL_BASE_DATE_PUBLISHED	Publication date (date when it become available for subscription)	SignalBaseGetInteger
SIGNAL_BASE_DATE_STARTED	Monitoring starting date	SignalBaseGetInteger
SIGNAL_BASE_EQUITY	Account equity	SignalBaseGetDouble
SIGNAL_BASE_GAIN	Account gain	SignalBaseGetDouble
SIGNAL_BASE_ID	Signal ID	SignalBaseGetInteger
SIGNAL_BASE_LEVERAGE	Hesap kaldıracı	SignalBaseGetInteger
SIGNAL_BASE_MAX_DRAWDOWN	Account maxi	SignalBaseGetDouble

Sabit	Açıklama	Usage
	mum drawdown	
SIGNAL_BASE_NAME	Signal name	SignalBaseGetString
SIGNAL_BASE_PIPS	Profit in pips	SignalBaseGetInteger
SIGNAL_BASE_PRICE	Signal subscription price	SignalBaseGetDouble
SIGNAL_BASE_RATING	Position in rating	SignalBaseGetInteger
SIGNAL_BASE_ROI	Return on Investment (%)	SignalBaseGetDouble
SIGNAL_BASE_SUBSCRIBERS	Number of subscribers	SignalBaseGetInteger
SIGNAL_BASE_TRADE_MODE	Account type (0-real, 1-demo, 2-contest)	SignalBaseGetInteger
SIGNAL_BASE_TRADES	Number of trades	SignalBaseGetInteger
SIGNAL_INFO_CONFIRMATIONS_DISABLED	The flag enables synchr	SignalInfoGetInteger, SignalInfoSetInteger

Sabit	Açıklama	Usage
	onization without confirmation dialog	
SIGNAL_INFO_COPY_SLTP	Copy Stop Loss and Take Profit flag	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_DEPOSIT_PERCENT	Deposit percent (%)	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_EQUITY_LIMIT	Equity limit	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_INFO_ID	Signal id, r/o	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_NAME	Signal name, r/o	SignalInfoGetString
SIGNAL_INFO_SLIPPAGE	Slippage (used when placing market orders in synchronization of positions and copyin	SignalInfoGetDouble, SignalInfoSetDouble

Sabit	Açıklama	Usage
	g of trades)	
SIGNAL_INFO_SUBSCRIPTION_ENABLED	"Copy trades by subscription" permission flag	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_TERMS_AGREE	"Agree to terms of use of Signals service" flag, r/o	SignalInfoGetInteger, SignalInfoSetInteger
SIGNAL_INFO_VOLUME_PERCENT	Maximum percent of deposit used (%), r/o	SignalInfoGetDouble, SignalInfoSetDouble
SIGNAL_LINE	Sinyal çizgisi	Gösterge Çizgileri
STAT_BALANCE_DD	Parasal terimlerle bakiyedeki maksimum azalma. Alım-	TesterStatistics

Sabit	Açıklama	Usage
	satım süreci içinde bakiyede birçok azalma gerçekleşebilir; burada en büyük değer alınır	
STAT_BALANCE_DD_RELATIVE	Parasal terimlerle bakiyedeki azalma, bakiyedeki maksimum yüzdelik azalmanın (STAT_BALANCE_DDREL_PERCENT) gerçekleşmesiyle kaydedilir.	TesterStatistics
STAT_BALANCE_DDREL_PERCENT	Yüzdelik olarak	TesterStatistics

Sabit	Açıklama	Usage
	bakiyedeki maksimum azalma. Alım-satım süreci içinde bakiyede birçok azalma gerçekleşebilir, burada her biri için yüzdelik bazda göreli azalma değeri hesaplanır. En büyük değere dönüş yapılır	
STAT_BALANCEDD_PERCENT	Yüzdelik değerle bakiyedeki azalma, bakiyedeki	TesterStatistics

Sabit	Açıklama	Usage
	maksimum azalmanın (STAT_BALANCE_DD) gerçekleşmesiyle kaydedilir.	
STAT_BALANCEMIN	Minimum bakiye değeri	TesterStatistics
STAT_CONLOSSMAX	Bir kayıplı işlem serisi içindeki maksimum zarar. Değer, sıfırdan küçük tür veya sıfıra eşittir	TesterStatistics
STAT_CONLOSSMAX_TRADES	STAT_CONLOSSMAX (bir kaybeden işlem serisi	TesterStatistics

Sabit	Açıklama	Usage
	içindeki maksimum zarar) istatistiğini şekillendiren işlemlerin sayısı	
STAT_CONPROFITMAX	Bir kazançlı işlem serisi içindeki maksimum kar. Değer, sıfırdan büyük tür veya sıfıra eşittir	TesterStatistics
STAT_CONPROFITMAX_TRADES	STAT_CONPROFITMAX (bir kazançlı işlem serisi içindeki maksimum kar) istatis	TesterStatistics

Sabit	Açıklama	Usage
	tiğini şekillendiren işlemlerin sayısı	
STAT_CUSTOM_ONTESTER	OnTester() fonksiyonunun dönüş yaptığı, hesaplanmış özel optimizasyon kriterinin değeri	TesterStatistics
STAT_DEALS	İşlemlerin sayısı	TesterStatistics
STAT_EQUITY_DD	Paralterimlerle varlıktaki maksimum azalma. Alım-satım süreci içinde varlıkta birçok azalma gerçe	TesterStatistics

Sabit	Açıklama	Usage
	kleşebilir; burada en büyük değer alınmaktadır	
STAT_EQUITY_DD_RELATIVE	Parasal terimlerle varlıktaki azalma, varlıktaki maksimum yüzdelik azalma (STAT_EQUITY_DDREL_PERCENT) gerçekleştiğinde kaydedilir.	TesterStatistics
STAT_EQUITY_DDREL_PERCENT	Varlıktaki maksimum yüzdelik azalma. Alım-satım	TesterStatistics

Sabit	Açıklama	Usage
	süreci içinde varlıkta birçok azalma gerçekleşebilir, burada her biri için yüzdelik bazda, göreli azalma değeri hesaplanır. En büyük değere dönüş yapılır	
STAT_EQUITYDD_PERCENT	Yüzdelik olarak azalma miktarı, varlıktaki maksimum paras al azalmanın (STAT_EQUI	TesterStatistics

Sabit	Açıklama	Usage
	TY_DD) gerçekleşmesiyle kaydedilir.	
STAT_EQUITYMIN	Minimum varlık değeri	TesterStatistics
STAT_EXPECTED_PAYOFF	Beklenen kazanç	TesterStatistics
STAT_GROSS_LOSS	Toplam kayıp, tüm negatif alım-satım işlemlerinin toplamı. Değer, sıfırdan küçük tür veya sıfıra eşittir	TesterStatistics
STAT_GROSS_PROFIT	Toplam kar, tüm kazançlı (pozitif) alım-	TesterStatistics

Sabit	Açıklama	Usage
	satım işlemlerinin toplamı. Değer, sıfırdan büyük tür veya sıfıra eşittir	
STAT_INITIAL_DEPOSIT	Başlangıç mevduat değeri	TesterStatistics
STAT_LONG_TRADES	Uzun (alış) işlemler	TesterStatistics
STAT_LOSS_TRADES	Kaybeden işlemler	TesterStatistics
STAT_LOSSTRADES_AVGCON	Kaybeden bir işlem serisinin ortalama uzunluğu	TesterStatistics
STAT_MAX_CONLOSS_TRADES	En uzun kaybeden işlem serisindeki işlem	TesterStatistics

Sabit	Açıklama	Usage
	erin sayısı STAT_MAX_CONLOSSES	
STAT_MAX_CONLOSSES	En uzun kaybeden işlem serisindeki toplam zarar	TesterStatistics
STAT_MAX_CONPROFIT_TRADES	En uzun kazançlı işlemler serisindeki işlem sayısı STAT_MAX_CONWINS	TesterStatistics
STAT_MAX_CONWINS	En uzun kazançlı işlemler serisinin toplam karı	TesterStatistics
STAT_MAX_LOSSTRADE	Maksimum kayıp	TesterStatistics

Sabit	Açıklama	Usage
	- tüm kaybeden işlemlerin arasındaki en küçük değer. Değer, sıfırdan küçük tür veya sıfıra eşittir	
STAT_MAX_PROFITTRADE	Maksimum kar - tüm kazançlı işlemler içindeki en büyük değer. Değer, sıfırdan büyük tür veya sıfıra eşittir	TesterStatistics
STAT_MIN_MARGINLEVEL	Minimum teminat	TesterStatistics

Sabit	Açıklama	Usage
	seviyesi değeri	
STAT_PROFIT	Sınamadan sonraki net kar, STAT_GROSS_PROFIT ve STAT_GROSS_LOSS değerlerinin toplamı (STAT_GROSS_LOSS her zaman sıfırdan küçük veya sıfıra eşittir)	TesterStatistics
STAT_PROFIT_FACTOR	Kazanç faktörü, STAT_GROSS_PROFIT/STAT_GROSS_LOSS	TesterStatistics

Sabit	Açıklama	Usage
	oranına eşittir. Eğer STAT_GROSS_LOSS=0 ise, kazanç faktörü DBL_MAX değerine eşittir	
STAT_PROFIT_LONGTRADES	Kazançlı uzun işlemler	TesterStatistics
STAT_PROFIT_SHORTTRADES	Kazançlı kısa işlemler	TesterStatistics
STAT_PROFIT_TRADES	Kazançlı işlemler	TesterStatistics
STAT_PROFITTRADES_AVGCON	Kazançlı bir işlem serisinin ortalama uzunluğu	TesterStatistics
STAT_RECOVERY_FACTOR	Geri kazanım faktör	TesterStatistics

Sabit	Açıklama	Usage
	ü, STAT_PROFIT/STAT_BALANCE_DD oranına eşittir	
STAT_SHARPE_RATIO	Sharpe oranı	TesterStatistics
STAT_SHORT_TRADES	Kısa (satış) işlemler	TesterStatistics
STAT_TRADES	Yapılan işlemlerinin sayısı	TesterStatistics
STAT_WITHDRAWAL	Hesaptan çekilen para miktarı	TesterStatistics
STO_CLOSECLOSE	Close/Close fiyatları temelinde hesaplama	Fiyat Sabitleri
STO_LOWHIGH	Low/High fiyatları temeli	Fiyat Sabitleri

Sabit	Açıklama	Usage
	nde hesaplama	
STYLE_DASH	Kesikli çizgi	Çizim Stilleri
STYLE_DASHDOT	Kesikli-noktalı çizgi	Çizim Stilleri
STYLE_DASHDOTDOT	Kesikli - çift noktalı çizgi	Çizim Stilleri
STYLE_DOT	Noktalı çizgi	Çizim Stilleri
STYLE_SOLID	Düz çizgi	Çizim Stilleri
SUNDAY	Pazar	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
SYMBOL_ASK	Ask - en iyi alım teklifi	SymbolInfoDouble
SYMBOL_ASKHIGH	Günün en yüksek Alış değeri	SymbolInfoDouble
SYMBOL_ASKLOW	Günün en düşük Alış değeri	SymbolInfoDouble
SYMBOL_BANK	Mevcut fiyat teklifinin dağıtıcısı	SymbolInfoString

Sabit	Açıklama	Usage
SYMBOL_BASIS	The underlying asset of a derivative	SymbolInfoString
SYMBOL_BID	Bid - en iyi satış teklifi	SymbolInfoDouble
SYMBOL_BIDHIGH	Günün en yüksek Satış değeri	SymbolInfoDouble
SYMBOL_BIDLOW	Günün en düşük Satış değeri	SymbolInfoDouble
SYMBOL_CALC_MODE_CFD	CFD mode - CFD için kar ve teminat değerlerinin hesaplanması	SymbolInfoInteger
SYMBOL_CALC_MODE_CFDINDEX	CFD index mode - CFD için kar ve teminat değerlerinin endek	SymbolInfoInteger

Sabit	Açıklama	Usage
	slerle hesaplanması	
SYMBOL_CALC_MODE_CFDLEVERAGE	CFD Leverage mode - CFD'de kaldıraçlı alım-satım için için kar ve teminat değerlerinin hesaplanması	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_FUTURES	Futures mode - hisse senedi piyasasında vadeli işlem sözleşmelerinin alım-satımı için kar ve teminat değerl	SymbolInfoInteger

Sabit	Açıklama	Usage
	erinin hesaplanması	
SYMBOL_CALC_MODE_EXCH_FUTURES_FORTS	FORTS Futures mode - FORTS üzerinde vadeli işlem sözleşmelerinin alım-satımı için kar ve teminat değerlerinin hesaplanması.	SymbolInfoInteger
SYMBOL_CALC_MODE_EXCH_STOCKS	Exchange mode - hisse senedi piyasalarında, menkul değer alım-satımı için	SymbolInfoInteger

Sabit	Açıklama	Usage
	kar ve teminat değerlerinin hesaplanması	
SYMBOL_CALC_MODE_FOREX	Forex mode - Forex için kar ve teminat değerlerinin hesaplanması	SymbolInfoInteger
SYMBOL_CALC_MODE_FUTURES	Futures mode - vadeli ürünler için kar ve teminat değerlerinin hesaplanması	SymbolInfoInteger
SYMBOL_CALC_MODE_SERV_COLLATERAL	Collateral mode - a symbol is used as a non-trada	SymbolInfoInteger

Sabit	Açıklama	Usage
	ble asset on a trading account.	
SYMBOL_CURRENCY_BASE	Sembolün baz dövizi	SymbolInfoString
SYMBOL_CURRENCY_MARGIN	Teminat döviz birimi	SymbolInfoString
SYMBOL_CURRENCY_PROFIT	Kar dövizi	SymbolInfoString
SYMBOL_DESCRIPTION	Sembol açıklaması	SymbolInfoString
SYMBOL_DIGITS	Ondalık noktadan sonraki basamak sayısı	SymbolInfoInteger
SYMBOL_EXPIRATION_DAY	Emir gün sonuna kadar geçerlidir	SymbolInfoInteger
SYMBOL_EXPIRATION_GTC	Açık bir şekilde iptal edilmediği	SymbolInfoInteger

Sabit	Açıklama	Usage
	süreç, emir sınırsız bir zaman aralığı için geçerlidir.	
SYMBOL_EXPIRATION_MODE	İzin verilen emir zaman-aşımı modlarının bayrakları	SymbolInfoInteger
SYMBOL_EXPIRATION_SPECIFIED	Zaman-aşımı zamanı emirde belirtilir	SymbolInfoInteger
SYMBOL_EXPIRATION_SPECIFIED_DAY	Zaman-aşımı tarihi emirde belirtilir	SymbolInfoInteger
SYMBOL_EXPIRATION_TIME	Sembol alım-satım bitiş tarihi	SymbolInfoInteger

Sabit	Açıklama	Usage
	(genellikle vadeli işlemlerde kullanılır)	
SYMBOL_FILLING_FOK	Bu emir türü, işlemin sadece belirtilen hacimle gerçekleştirilebileceği anlamına gelir. Eğer piyasada yeterli miktarda finansal enstrüman bulunmuyorsa, emir yürürlüğe konulmaz. İstenen hacim değeri	SymbolInfoInteger

Sabit	Açıklama	Usage
	, o an piyasada mevcut olan birkaç teklif ile karşılanabilir.	
SYMBOL_FILLING_IOC	Bu durumda kullanıcı, emirde belirtilen sınırlar içerisinde piyasada bulunan maksimum hacim değeriyle işlem yapmaya razı olur. Burada emir tamamen karşılanmayabilir	SymbolInfoInteger

Sabit	Açıklama	Usage
	, mevcut hacim karşılanacak ve kalanı iptal edilecektir. IOC emirlerinin kullanım durumu, alım-satım sunucusunda belirlenir.	
SYMBOL_FILLING_MODE	izin verilen emir uygulama modu bayrakları	SymbolInfoInteger
SYMBOL_ISIN	Bir sembolün ISIN (International Securities Identification	SymbolInfoString

Sabit	Açıklama	Usage
	Number) sistemindeki ismi. Uluslararası Menkul Değerleri Tanımlama Numarası, bir menkul değeri benzersiz şekilde tanımlayan 12-haneli bir alfa-sayısal koddur. Bu sembol özelliğinin varlığı, alım-satım sunucusu tarafında belirlenir.	

Sabit	Açıklama	Usage
SYMBOL_LAST	Son işlem fiyatı	SymbolInfoDouble
SYMBOL_LASTHIGH	Günün en yüksek Last değeri	SymbolInfoDouble
SYMBOL_LASTLOW	Günün en düşük Last değeri	SymbolInfoDouble
SYMBOL_MARGIN_INITIAL	Başlangıç teminatı, mevduat dövizi cinsinden bir lotluk pozisyon açmak için gerekli olan teminat miktarı anlamındadır. Bu, bir müşterinin, piyasaya girdiği zaman mal	SymbolInfoDouble

Sabit	Açıklama	Usage
	varlığının kontrol edilmesi için kullanılır.	
SYMBOL_MARGIN_MAINTENANCE	Sürdürme teminatı. Eğer ayarlanmışsa, teminat miktarını mevduat dövizi cinsinden ayarlar, bir lottan itibaren değer alır. Müşterinin mal varlığını, hesap durumu değiştiğinde kontrol etmek içi	SymbolInfoDouble

Sabit	Açıklama	Usage
	kullanılır. Sürdürme teminatının 0 olması durumunda, başlangıç teminatı kullanılır.	
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 option is specif	SymbolInfoInteger

Sabit	Açıklama	Usage
	ied 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	SymbolInfoInteger

Sabit	Açıklama	Usage
	right to sell an asset at a specified price	
SYMBOL_OPTION_STRIKE	The strike price of an 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 amou	SymbolInfoDouble

Sabit	Açıklama	Usage
	nt of the underlying asset	
SYMBOL_ORDER_LIMIT	Limitli emirlere izin verilir (Buy Limit ve Sell Limit)	SymbolInfoInteger
SYMBOL_ORDER_MARKET	Piyasa emirlerine izin verilir (Buy ve Sell)	SymbolInfoInteger
SYMBOL_ORDER_MODE	İzin verilen emir tipi bayrakları	SymbolInfoInteger
SYMBOL_ORDER_SL	Stop Loss (zararı durdur) emrine izin verilir	SymbolInfoInteger
SYMBOL_ORDER_STOP	Stop (durdurma) emirlerine izin	SymbolInfoInteger

Sabit	Açıklama	Usage
	verilir (Buy Stop ve Sell Stop)	
SYMBOL_ORDER_STOP_LIMIT	Stop-limit emirlerine izin verilir (Buy Stop Limit ve Sell Stop Limit)	SymbolInfoInteger
SYMBOL_ORDER_TP	Take Profit (kar al) emrine izin verilir	SymbolInfoInteger
SYMBOL_PATH	Sembol ağacındaki veri adresi	SymbolInfoString
SYMBOL_POINT	Sembol puan değeri	SymbolInfoDouble
SYMBOL_SELECT	Piyasa Gözlemi içinde seçilen sembol	SymbolInfoInteger

Sabit	Açıklama	Usage
SYMBOL_SESSION_AW	Mevcut seansın ağırlıklı ortalama fiyatı	SymbolInfoDouble
SYMBOL_SESSION_BUY_ORDERS	Mevcut Alım emirlerinin sayısı	SymbolInfoInteger
SYMBOL_SESSION_BUY_ORDERS_VOLUME	Alım emirlerinin mevcut hacmi	SymbolInfoDouble
SYMBOL_SESSION_CLOSE	Mevcut seansın kapanış fiyatı	SymbolInfoDouble
SYMBOL_SESSION_DEALS	Mevcut seans içindeki işlemlerin sayısı	SymbolInfoInteger
SYMBOL_SESSION_INTEREST	Açık vadeli işlemlerin özeti	SymbolInfoDouble
SYMBOL_SESSION_OPEN	Mevcut seansın	SymbolInfoDouble

Sabit	Açıklama	Usage
	açılış fiyatı	
SYMBOL_SESSION_PRICE_LIMIT_MAX	Mevcut seansın en yüksek fiyatı	SymbolInfoDouble
SYMBOL_SESSION_PRICE_LIMIT_MIN	Mevcut seansın en düşük fiyatı	SymbolInfoDouble
SYMBOL_SESSION_PRICE_SETTLEMENT	Mevcut seanstaki uzlaşma fiyatı	SymbolInfoDouble
SYMBOL_SESSION_SELL_ORDERS	Mevcut Satış emirlerinin sayısı	SymbolInfoInteger
SYMBOL_SESSION_SELL_ORDERS_VOLUME	Satış emirlerinin mevcut hacmi	SymbolInfoDouble
SYMBOL_SESSION_TURNOVER	Mevcut seans cirosunun özeti	SymbolInfoDouble
SYMBOL_SESSION_VOLUME	Mevcut seans	SymbolInfoDouble

Sabit	Açıklama	Usage
	pozisyonlarının özet hacim değeri	
SYMBOL_SPREAD	Puan bazında makas değeri	SymbolInfoInteger
SYMBOL_SPREAD_FLOAT	Değişken makas gösterimi	SymbolInfoInteger
SYMBOL_START_TIME	Sembol alım-satım başlangıç tarihi (genellikle vadeli işlemlerde kullanılır)	SymbolInfoInteger
SYMBOL_SWAP_LONG	Uzun swap değeri	SymbolInfoDouble
SYMBOL_SWAP_MODE	Swap hesaplama modeli	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_DEPOSIT	Swap bedelleri, müşte	SymbolInfoInteger

Sabit	Açıklama	Usage
	rinin teminat para birimi üzerinden uygulanır	
SYMBOL_SWAP_MODE_CURRENCY_MARGIN	Swap bedelleri semboldeki karşıt döviz üzerinden uygulanır	SymbolInfoInteger
SYMBOL_SWAP_MODE_CURRENCY_SYMBOL	Swap bedelleri semboldeki baz döviz birimi üzerinden uygulanır	SymbolInfoInteger
SYMBOL_SWAP_MODE_DISABLED	Swap devre dışı (swap yok)	SymbolInfoInteger
SYMBOL_SWAP_MODE_INTEREST_CURRENT	Swap bedelleri, swap hesabı sırasında	SymbolInfoInteger

Sabit	Açıklama	Usage
	finansal enstrüman fiyatının belirlenen yıllık faizi üzerinden uygulanır (standart bankayılı 360 gündür)	
SYMBOL_SWAP_MODE_INTEREST_OPEN	Swap bedelleri, pozisyonun açılış fiyatının belirlenen yıllık faizi üzerinden uygulanır (standart bankayılı 360 gündür)	SymbolInfoInteger
SYMBOL_SWAP_MODE_POINTS	Swap bedell	SymbolInfoInteger

Sabit	Açıklama	Usage
	eri puan bazında uygulanır	
SYMBOL_SWAP_MODE_REOPEN_BID	Swap bedelleri yeniden açılan pozisyonlar üzerinden uygulanır. Alım-satım gününün sonunda pozisyon kapatılır. Ertesi gün, mevcut Satış fiyatı +/- belirlenen puan sayısı (SYMBOL_SWAP_LONG ve SYMBOL_SWAP_	SymbolInfoInteger

Sabit	Açıklama	Usage
	SHORT parametreleri) ile yeniden açılır	
SYMBOL_SWAP_MODE_REOPEN_CURRENT	Swap bedelleri yeniden açılan pozisyonlar üzerinden uygulanır. Alım-satım gününün sonunda pozisyon kapatılır. Ertesi gün pozisyonlar, kapanış fiyatı +/- belirlenen puan sayısı (SYMBOL_SWAP_LONG ve	SymbolInfoInteger

Sabit	Açıklama	Usage
	SYMBOL_SWAP_SHORT parametreleri) ile yeniden açılır	
SYMBOL_SWAP_ROLLOVER3DAYS	Hafta içinde, 3 günlük swap değerinin kesildiği gün	SymbolInfoInteger
SYMBOL_SWAP_SHORT	Kısa swap değeri	SymbolInfoDouble
SYMBOL_TICKS_BOOKDEPTH	Piyasa derinliğinde gösterilen isteklerin maksimal sayısı. İstek kuyruğu bulunmayan sembollerde, bu değer	SymbolInfoInteger

Sabit	Açıklama	Usage
	varsayılan olarak sıfırdır.	
SYMBOL_TIME	Son fiyat teklifinin zamanı	SymbolInfoInteger
SYMBOL_TRADE_CALC_MODE	İşlem fiyatı hesaplama modu	SymbolInfoInteger
SYMBOL_TRADE_CONTRACT_SIZE	Alım-satım sözleşme büyüklüğü	SymbolInfoDouble
SYMBOL_TRADE_EXECUTION_EXCHANGE	Borsa uygulaması	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_INSTANT	Anında uygulama	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_MARKET	Piyasa Uygulaması	SymbolInfoInteger
SYMBOL_TRADE_EXECUTION_REQUEST	İstek ile uygulama	SymbolInfoInteger
SYMBOL_TRADE_EXEMODE	İşlem gerçekleştirme modu	SymbolInfoInteger

Sabit	Açıklama	Usage
SYMBOL_TRADE_FREEZE_LEVEL	Puan bazında, alım-satım işlemlerinin dondurulmasına olan uzaklık	SymbolInfoInteger
SYMBOL_TRADE_MODE	Emir gerçekleştirme tipi	SymbolInfoInteger
SYMBOL_TRADE_MODE_CLOSEONLY	Sadece pozisyon kapama işlemlerine izin verilir	SymbolInfoInteger
SYMBOL_TRADE_MODE_DISABLED	Sembol için alım-satım devre dışı bırakılmıştır	SymbolInfoInteger
SYMBOL_TRADE_MODE_FULL	Alım-satım kısıtlaması yoktur	SymbolInfoInteger
SYMBOL_TRADE_MODE_LONGONLY	Sadece uzun pozisy	SymbolInfoInteger

Sabit	Açıklama	Usage
	onlara izin verilir	
SYMBOL_TRADE_MODE_SHORTONLY	Sadece kısa pozisyonlara izin verilir	SymbolInfoInteger
SYMBOL_TRADE_STOPS_LEVEL	Stop emrinin verilebilmesi için, mevcut kapanış fiyatından itibaren, puan bazında gerekli minimum fark	SymbolInfoInteger
SYMBOL_TRADE_TICK_SIZE	En düşük fiyat değişimi	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE	SYMBOL_TRADE_TICK_VALUE_PROFIT değeri	SymbolInfoDouble

Sabit	Açıklama	Usage
SYMBOL_TRADE_TICK_VALUE_LOSS	Kaybetmekte olan bir pozisyon için hesaplanan tik fiyatı	SymbolInfoDouble
SYMBOL_TRADE_TICK_VALUE_PROFIT	Karlı olabilecek bir pozisyon için hesaplanan tik fiyatı	SymbolInfoDouble
SYMBOL_VOLUME	Son işlemin hacmi	SymbolInfoInteger
SYMBOL_VOLUME_LIMIT	Bir semboldeki tek yönlü (alış veya satış) açık emirler ve bekleyen emirler için izin verilen maksimum	SymbolInfoDouble

Sabit	Açıklama	Usage
	toplam hacim. Örneğin 5 lotluk bir sınırlama ile, 5 lotluk hacimde açık bir alım pozisyonuna sahip olabilir ve yine 5 lotluk hacimde bir Sell Limit bekleyen emri girebilirsiniz. Ama bu durumda (tek yöndeki toplam hacim, belirlenen sınırlamayı	

Sabit	Açıklama	Usage
	geçemeyeceğinden) bir Buy Limit bekleyen emri veya 5 lottan büyük bir Sell Limit emri girmezsiniz.	
SYMBOL_VOLUME_MAX	Bir işlemde kullanılabilecek en yüksek hacim	SymbolInfoDouble
SYMBOL_VOLUME_MIN	Bir işlemde kullanılabilecek en düşük hacim	SymbolInfoDouble
SYMBOL_VOLUME_STEP	Bir işlemde kullanılabilecek	SymbolInfoDouble

Sabit	Açıklama	Usage
	en düşük hacim değişim birimi	
SYMBOL_VOLUMEHIGH	Günün en yüksek hacim değeri	SymbolInfoInteger
SYMBOL_VOLUMELOW	Günün en düşük hacim değeri	SymbolInfoInteger
TENKANSEN_LINE	Tenkan-sen çizgisi	Gösterge Çizgileri
TERMINAL_BUILD	Müşteri terminalinin kurulum numarası	TerminalInfoInteger
TERMINAL_CODEPAGE	Müşteri terminalinde yüklü olan dilin kod sayfasının numarası	TerminalInfoInteger
TERMINAL_COMMONDATA_PATH	Bilgis ayara	TerminalInfoString

Sabit	Açıklama	Usage
	yüklenen tüm terminaller için çoğunlukla kullanılan dosya adresi	
TERMINAL_COMMUNITY_ACCOUNT	The flag indicates the presence of MQL5.community authorization data in the terminal	TerminalInfoInteger
TERMINAL_COMMUNITY_BALANCE	Balance in MQL5.community	TerminalInfoDouble
TERMINAL_COMMUNITY_CONNECTION	Connection to MQL5.community	TerminalInfoInteger
TERMINAL_COMPANY	Firma ismi	TerminalInfoString

Sabit	Açıklama	Usage
TERMINAL_CONNECTED	Bir alım-satım sunucusuna yapılan bağlantı	TerminalInfoInteger
TERMINAL_CPU_CORES	Sistemdeki CPU (işlemci) çekirdeklerinin sayısı	TerminalInfoInteger
TERMINAL_DATA_PATH	Terminal verisinin saklandığı klasör	TerminalInfoString
TERMINAL_DISK_SPACE	Terminal biriminin MQL5\Files klasörü için kullanılabilecek serbest disk alanı, MB	TerminalInfoInteger
TERMINAL_DLLS_ALLOWED	DLL kullanım izni	TerminalInfoInteger

Sabit	Açıklama	Usage
TERMINAL_EMAIL_ENABLED	SMTP sunucusu ve terminal ayarlarında belirtilmiş giriş bilgisini (login) kullanılarak, e-mail gönderme izni	TerminalInfoInteger
TERMINAL_FTP_ENABLED	FTP sunucusu ve terminal ayarlarında belirtilmiş giriş bilgisini (login) kullanılarak, rapor gönderme izni	TerminalInfoInteger
TERMINAL_LANGUAGE	Terminal	TerminalInfoString

Sabit	Açıklama	Usage
	dili	
TERMINAL_MAXBARS	Çizelge üzerindeki maksimum çubuk sayısı	TerminalInfoInteger
TERMINAL_MEMORY_AVAILABLE	Terminal birimi işlemlerindeki serbest bellek, MB	TerminalInfoInteger
TERMINAL_MEMORY_PHYSICAL	Sistemdeki mevcut fiziksel bellek, MB	TerminalInfoInteger
TERMINAL_MEMORY_TOTAL	Terminal birimi işlemleri için kullanılabilir bellek büyüklüğü, MB	TerminalInfoInteger
TERMINAL_MEMORY_USED	Temsilci tarafından kullanılan	TerminalInfoInteger

Sabit	Açıklama	Usage
	bellek miktarı, MB	
TERMINAL_MQID	The flag indicates the presence of MetaQuotes ID data to send Push notifications	TerminalInfoInteger
TERMINAL_NAME	Terminal ismi	TerminalInfoString
TERMINAL_NOTIFICATIONS_ENABLED	Permission to send notifications to smartphone	TerminalInfoInteger
TERMINAL_OPENCL_SUPPORT	Desteklenen OpenCL versiyonu, 0x00010002 = 1.2. biçiminde verilir	TerminalInfoInteger

Sabit	Açıklama	Usage
	"0" sayısı OpenCL'in desteklenmediği anlamına gelir	
TERMINAL_PATH	Terminalin başlatıldığı klasör	TerminalInfoString
TERMINAL_PING_LAST	Alım-satım sunucusuna gönderilen son ping sinyalinin mikrosaniye cinsinden değeri. Bir saniye, bir milyon mikrosaniyeden oluşur	TerminalInfoInteger
TERMINAL_SCREEN_DPI	Ekrandaki görüntü çözünürlüğü bir inçlik	TerminalInfoInteger

Sabit	Açıklama	Usage
	çizgi üzerindeki nokta sayısıyla (DPI - Dots per Inch) ölçülür	
TERMINAL_TRADE_ALLOWED	Alım-satım yapma izni	TerminalInfoInteger
TERMINAL_X64	"64-bitlik terminal" gösterimi	TerminalInfoInteger
THURSDAY	Perşembe	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
TRADE_ACTION_DEAL	Anlık uygulama için belirlenen parametrelerle bir alım-satım emrini işleme koyar (piyasa emri)	MqlTradeRequest
TRADE_ACTION_MODIFY	Daha önce verilm	MqlTradeRequest

Sabit	Açıklama	Usage
	iş bir emrin parametrelerini şekillendirir	
TRADE_ACTION_PENDING	Bir alım-satım emrini, belirtilen koşullar altında uygulanacak şekilde işleme koyar (bekleyen emir)	MqlTradeRequest
TRADE_ACTION_REMOVE	Önceden verilmiş bekleyen emri siler	MqlTradeRequest
TRADE_ACTION_SLTP	Bir açık pozisyon için Zarar Durdur ve Kar Al değerl	MqlTradeRequest

Sabit	Açıklama	Usage
	erini şekillendirir	
TRADE_RETCODE_CANCEL	İstek kullanıcı tarafından iptal edildi	MqlTradeResult
TRADE_RETCODE_CLIENT_DISABLES_AT	Otomatik alım-satım, müşteri terminali tarafından devre dışı bırakıldı	MqlTradeResult
TRADE_RETCODE_CONNECTION	Alım-satım sunucusuyla bağlantı yok	MqlTradeResult
TRADE_RETCODE_DONE	İstek tamamlandı	MqlTradeResult
TRADE_RETCODE_DONE_PARTIAL	İsteğin sadece bir kısmı tamamlandı	MqlTradeResult

Sabit	Açıklama	Usage
TRADE_RETCODE_ERROR	İstek işleme hatası	MqlTradeResult
TRADE_RETCODE_FROZEN	Emir veya pozisyon dondurulmuş	MqlTradeResult
TRADE_RETCODE_INVALID	Geçersiz istek	MqlTradeResult
TRADE_RETCODE_INVALID_EXPIRATION	İstek içinde geçersiz zaman-aşımı tarihi	MqlTradeResult
TRADE_RETCODE_INVALID_FILL	Geçersiz emir işleme tipi	MqlTradeResult
TRADE_RETCODE_INVALID_ORDER	Yanlış veya geçersiz emir tipi	MqlTradeResult
TRADE_RETCODE_INVALID_PRICE	Geçersiz istek fiyatı	MqlTradeResult
TRADE_RETCODE_INVALID_STOPS	İstek içinde geçersiz stop (durd	MqlTradeResult

Sabit	Açıklama	Usage
	urma) seviyeleri	
TRADE_RETCODE_INVALID_VOLUME	Geçersiz istek hacmi	MqlTradeResult
TRADE_RETCODE_LIMIT_ORDERS	Bekleyen emirlerin sayısı sınıra ulaştı	MqlTradeResult
TRADE_RETCODE_LIMIT_VOLUME	Sembol üzerindeki emirler ve pozisyonlar için hacim sınırına ulaşıldı	MqlTradeResult
TRADE_RETCODE_LOCKED	İstek işlem için kilitlendi	MqlTradeResult
TRADE_RETCODE_MARKET_CLOSED	Piyasa kapalı	MqlTradeResult
TRADE_RETCODE_NO_CHANGES	İstekte değişim yok	MqlTradeResult
TRADE_RETCODE_NO_MONEY	İsteği tamamlayabilece	MqlTradeResult

Sabit	Açıklama	Usage
	k yeterli para yok	
TRADE_RETCODE_ONLY_REAL	Sadece gerçek hesaplarda işleme izin veriliyor	MqlTradeResult
TRADE_RETCODE_ORDER_CHANGED	Emir durumu değişti	MqlTradeResult
TRADE_RETCODE_PLACED	Emir işlendi	MqlTradeResult
TRADE_RETCODE_POSITION_CLOSED	Belirtilmiş POSITION_IDENTIFIER özelliğine sahip olan pozisyon kapatılmış	MqlTradeResult
TRADE_RETCODE_PRICE_CHANGED	Fiyatlar değişti	MqlTradeResult
TRADE_RETCODE_PRICE_OFF	İstek içinde işlene bilece	MqlTradeResult

Sabit	Açıklama	Usage
	k bir fiyat teklifi yok	
TRADE_RETCODE_REJECT	İstek reddedildi	MqlTradeResult
TRADE_RETCODE_REQUOTE	Yeniden fiyatlandırma (requote)	MqlTradeResult
TRADE_RETCODE_SERVER_DISABLES_AT	Otomatik alım-satım, sunucu tarafından devre dışı bırakıldı	MqlTradeResult
TRADE_RETCODE_TIMEOUT	İstek zaman-aşımı nedeniyle iptal edildi	MqlTradeResult
TRADE_RETCODE_TOO_MANY_REQUESTS	İsteklerin sıklığı fazla	MqlTradeResult
TRADE_RETCODE_TRADE_DISABLED	Alım-satım işlemleri	MqlTradeResult

Sabit	Açıklama	Usage
	devre dışı	
TRADE_TRANSACTION_DEAL_ADD	Geçmişe bir işlem eklenmesi. Bu eylem, bir emrin işlenmesinin veya hesap bakiyesi ile işlemler gerçekleştirilmesinin bir sonucu olarak ortaya çıkar.	MqlTradeTransaction
TRADE_TRANSACTION_DEAL_DELETE	Geçmişteki bir işlemin silinmesi. Daha önce gerçekleştirilen bir işlemin sunuc	MqlTradeTransaction

Sabit	Açıklama	Usage
	u üzerinde silinme olasılığı bulunmaktadır. Örneğin, işlem daha önce aracı kurum tarafından aktarıldığı bir dışsal alım-satım sisteminde (borsa) silinmiş olabilir.	
TRADE_TRANSACTION_DEAL_UPDATE	Geçmişteki bir işlemin güncellenmesi. Daha önce gerçekleştirilen bir	MqlTradeTransaction

Sabit	Açıklama	Usage
	işlemin, sunucu üzerinde değişme olasılığı bulunmaktadır. Örneğin, işlem daha önce aracı kurum tarafından aktarıldığı bir dışsal alım-satım sisteminde (borsa) değiştirilmiş olabilir.	
TRADE_TRANSACTION_HISTORY_ADD	Bir işlemin veya iptalin sonucu olarak bir emrin geçmi	MqlTradeTransaction

Sabit	Açıklama	Usage
	şe eklenmesi.	
TRADE_TRANSACTION_HISTORY_DELETE	Emir geçmişinden bir emrin silinmesi. Bu tip, sunucu tarafındaki fonksiyonelliğin artırılması için tasarlanmıştır.	MqlTradeTransaction
TRADE_TRANSACTION_HISTORY_UPDATE	Emir geçmişinde yer alan bir emrin değişmesi. Bu tip, sunucu tarafındaki fonksiyonelliğin artırılması için	MqlTradeTransaction

Sabit	Açıklama	Usage
	tasarlanmıştır.	
TRADE_TRANSACTION_ORDER_ADD	Yeni bir açık emir eklemesi.	MqlTradeTransaction
TRADE_TRANSACTION_ORDER_DELETE	Açık emirler listesinden bir emir silme. Bir emir, uygun bir isteğin ayarlanmasının veya uygulanmasının (karşılanmasının) ve geçmişe konulmasının bir sonucu olarak, açık emirler listesi	MqlTradeTransaction

Sabit	Açıklama	Usage
	nden siline bilir.	
TRADE_TRANSACTION_ORDER_UPDATE	Açılmış bir emri güncelle. Güncellemeler, sadece müşteri terminali veya alım-satım sunucusu tarafından yapılmış bariz değişimleri değil, aynı zamanda, emrin ayarlanması sırasında oluşan değişimleri de içerir (örneğin, ORDER_ST	MqlTradeTransaction

Sabit	Açıklama	Usage
	ATE_STARTED'dan ORDER_STATE_PLACED'a veya ORDER_STATE_PLACED'dan ORDER_STATE_PARTIAL'a vb. yapılan aktarımlar).	
TRADE_TRANSACTION_POSITION	İşlemle ilgili olmayan bir pozisyonun değiştirilmesi. Bu tür faaliyetler, bir pozisyonun sunucu tarafından değiştirildiğini	MqlTradeTransaction

Sabit	Açıklama	Usage
	göster ir. Pozisy on hacmi , açılış fiyatı, Zarar Durdu r ve Kar Al seviye leri değişt irilebil ir. Değişi mlere dair verile r, OnTra deTra nsacti on işleyic isi ile MqlTr adeTr ansac tion yapısı nda belirti lir. İşlemi n sonuc unda pozisy on değişi mi (ekle me, değişt irme	

Sabit	Açıklama	Usage
	veya kapama) olması, TRADE_TRANSACTION_POSITION faaliyetinin gerçekleşmesine yol açmaz.	
TRADE_TRANSACTION_REQUEST	Bir alım-satım isteğinin sunucu tarafından işlendiğine ve işlenen sonucunun alındığına dair uyarı. Bunun gibi faaliyetler için sadece MqlTr	MqlTradeTransaction

Sabit	Açıklama	Usage
	adeTransaction yapısının "type" alanı incelenmelidir. OnTradeTransaction fonksiyonunun ikinci ve üçüncü parametreleri (request ve result), ek veriler elde etmek için incelenmelidir.	
TUESDAY	Salı	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
TYPE_BOOL	bool	MqlParam
TYPE_CHAR	char	MqlParam
TYPE_COLOR	color	MqlParam
TYPE_DATETIME	datetime	MqlParam

Sabit	Açıklama	Usage
TYPE_DOUBLE	double	MqlParam
TYPE_FLOAT	float	MqlParam
TYPE_INT	int	MqlParam
TYPE_LONG	long	MqlParam
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	uchar tipi ile temsil edilebilecek maksimal değer	Nümerik Tip Sabitleri
UINT_MAX	uint tipi ile temsil edilebilecek maksimal değer	Nümerik Tip Sabitleri
ULONG_MAX	ulong tipi ile temsil edilebilecek maksimal değer	Nümerik Tip Sabitleri
UPPER_BAND	Üst limit	Gösterge Çizgileri

Sabit	Açıklama	Usage
UPPER_HISTOGRAM	Üst histogram	Gösterge Çizgileri
UPPER_LINE	Üst çizgi	Gösterge Çizgileri
USHORT_MAX	ushort tipi ile temsil edilebilecek maksimal değer	Nümerik Tip Sabitleri
VOLUME_REAL	Alım-satım hacmi	Fiyat Sabitleri
VOLUME_TICK	Tik hacmi	Fiyat Sabitleri
WEDNESDAY	Çarşamba	SymbolInfoInteger, SymbolInfoSessionQuote, SymbolInfoSessionTrade
WHOLE_ARRAY	Dizinin sonuna kadar var olan elemanın sayısıdır, yani bütün dizinin işleneceğini belirtir	Diğer Sabitler
WRONG_VALUE	Sabit, gizli	Diğer Sabitler

Sabit	Açıklama	Usage
	(örtülü) olarak herhangi bir sayım tipine dönüş türüle bilir	